Analysing Large-scale Surveillance Video

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy by

Yifan Zhou

Supervisors: Professor Simon Maskell and Professor Rahul Savani

Department of Computer Science
School of Electrical Engineering, Electronics and Computer Science

July 2018

Declaration

This dissertation is submitted for the degree of Doctor of Philosophy. I hereby certify that this dissertation constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions, or writings of another.

I declare that the dissertation describes original work that has not previously been presented for the award of any other degree of any institution.

To Zhang, Yi

Acknowledgements

I would like to thank my parents for their endless support not only during my PhD but the whole life. I am also grateful to Zhang Yi, who is always there when I meet any problems. This thesis cannot exist without your help.

I would like to thank Prof. Simon Maskell and Prof. Rahul Savani for being my supervisors. I am especially appreciated to Simon's guidance and advises during the entire PhD. I would also thank every member in Simon's group for the their help and creating the perfect academic atmosphere.

I would like to thank the department of computer science and the school of electrical engineering, electronics and computer science for fully funding my PhD. I would like to thank Simon again for extending the funding to four years, so I can achieve more.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acronyms

ATE	Absolute Trajectory Error
BRISK	Binary Robust Invariant Scalable Keypoints
DoF	Degree of Freedom
DLT	Direct Linear Transformation
EKF	Extended Kalman Filter
ESS	Effective Sample Size
GME	Global Motion Estimation
GM-PHD	Gaussian Mixture Probabilistic Hypothesis Density
MLESAC	Maximum Likelihood Estimation SAmple Consensus
NN	Nearest Neighbour
n-OP	near-Optimal Proposal
ORB	Oriented FAST and Rotated BRIEF
PDA	Probabilistic Data Association
PF	Particle Filter
PHD	Probabilistic Hypothesis Density
RANSAC	Random sample consensus
R-CNN	Regional-Convolutional Neural Networks
RB	Rao-Blackwellisation
RMSE	Root Mean Square Error
RPE	Relative Pose Error
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localisation And Mapping
SURF	Speeded Up Robust Features
SUT	Scaled Unscented Transformation
UKF	Unscented Kalman Filter
VO	Visual Odometry

Abstract

Analysing large-scale surveillance video has drawn significant attention because drone technology and high-resolution sensors are rapidly improving. The mobility of drones makes it possible to monitor a broad range of the environment, but it introduces a more difficult problem of identifying the objects of interest. This thesis aims to detect the moving objects (mostly vehicles) using the idea of background subtraction. Building a decent background is the key to success during the process. We consider two categories of surveillance videos in this thesis: when the scene is flat and when pronounced parallax exists. After reviewing several global motion estimation approaches, we propose a novel cost function, the log-likelihood of the student t-distribution, to estimate the background motion between two frames. The proposed idea enables the estimation process to be efficient and robust with auto-generated parameters. Since the particle filter is useful in various subjects, it is investigated in this thesis. An improvement to particle filters, combining near-optimal proposal and Rao-Blackwellisation, is discussed to increase the efficiency when dealing with non-linear problems. Such improvement is used to solve visual simultaneous localisation and mapping (SLAM) problems and we call it RB^2-PF. Its superiority is evident in both simulations of 2D SLAM and real datasets of visual odometry problems. Finally, RB^2-PF based visual odometry is the key component to detect moving objects from surveillance videos with pronounced parallax. The idea is to consider multiple planes in the scene to improve the background motion estimation. Experiments have shown that false alarms significantly reduced. With the landmark information, a ground plane can be worked out. A near-constant velocity model can be applied after mapping the detections on the ground plane regardless of the position and orientation of the camera. All the detection results are finally processed by a multi-target tracker, the Gaussian mixture probabilistic hypothesis density (GM-PHD) filter, to generate tracks.

Chapter 1

Introduction

1.1 Motivation

The objective of this thesis is to address the problems encountered when detecting and tracking moving objects from videos taken using moving cameras with a wide field of view. These cameras can be mounted on a drone and used to perform surveillance of large areas. Though the platform could be manned, the amount of data generated is so large that manual analysis is prohibitive in terms of time and effort. The particle filter is a useful tool in this context, both for tracking but also in a relevant technique called Simultaneous Localisation And Mapping (SLAM). Research in particle filtering can therefore be helpful in both contexts and can underpin an approach that tackles the moving-object detection problem in contexts involving pronounced parallax. This approach, and the posing of object detection as a foreground detection task, are evaluated during this PhD. Such foreground detection turns the detection problem into a number of classification problems. A popular algorithm of this kind is the Viola-Jones method [4, 5] which considers a sliding window and a classifier cascade. It is known that using sliding windows can be computational expensive. To address this, the classifier cascade involves a chain of fast and weak classifiers, such that most of the candidate target positions can be rejected very quickly. The chain is designed to also maximise precision, since for true targets to be detected, all the weak classifiers need to accept the candidate position. This approach has been investigated extensively in object detection (see, e.g., [6, 7]). However, since the approach involves a sliding window, it is very hard to make such an approach operate in real-time at relatively high resolutions using CPUs. Since Graphical Processing Units (GPUs) are widely available, it is now feasible to exploit the parallelisation of deep neural networks to perform image classification. Indeed there is current research (see e.g., [8, 9]) that involves the application of such deep learning techniques to image classification and achieves huge success. This encourages the research in object detection, such as [10] considers multi-scale windows and deep Convolutional Neural Networks based classifier. Subsequent development has centred on the Regional-Convolutional Neural Networks (R-CNN) [11], which aims to provide an efficient mechanism for proposing the positions of potential detections. Fast R-CNN [12] and Faster R-CNN [13] further reduce the runtime and make it possible to detect objects accurately and in real-time with videos. There are also other object detection approaches using deep learning (e.g., [14] and [15]). However, these

idea only necessarily focus on the instantaneous appearance of the objects and do not exploit the temporal information that presents in the video. The categories of objects that are detected also need to be pre-defined during the training stage. This makes it difficult to use such techniques to generate alerts when previously-unknown behaviours are encountered in surveillance videos. The focus of this thesis is on detecting moving objects from video with temporal information is on an unsupervised setting with an explicit focus on using the temporal information to overcome the issues mentioned above. The intent is that this will lay the foundations for future work that could focus on combining the temporal and appearance information to provide a more reliable analysis capability.

1.2 Contributions

Through this thesis, existing techniques were comprehensively discussed, new algorithms have been proposed and applied to solve real applications. The contributions are specified as follows.

- A robust student t-distribution is used to build a cost function for computing the global motion estimations from videos. The log-likelihood of the student t-distribution function acts the interpolation between the L2 norm and Cauchy function given different variables. By re-parameterising the log-likelihood function, the magnitude and the position of the peak of the cost function curve can be adjusted by each of the parameters individually. A parameter estimation method has also been proposed which trades off the efficiency and robustness while processing a video. Such contribution will be specified in Chapter 2.

- Rao-Blackwellisation and (near-)optimal proposal are two of the most significant techniques in particle filtering. This thesis discussed the combination of two considering correlated process noise. Several scenarios have been used to investigate when the combination of the two, or each one alone should be used in practise. This achievement will be described in Chapter 3.

- Dual Rao-Blackwellisation (called RB^2-PF in the following) and the near-optimal proposal is used for solving the problem of visual odometry from monocular camera. Different from the existing FastSLAM 2.0 which only uses Rao-Blackwellisation once to divide the camera state and landmark positions into two sub-problems, this approach further divides the camera state into two sub-problems, position and velocity, with the idea of Rao-Blackwellisation. It is novel that near-optimal proposal is applied while the monocular camera is the only input. Chapter 4 includes the contribution which has been published in [16].
  Y. Zhou and S. Maskell, "RB^2-PF : A novel filter-based monocular visual odometry algorithm," 2017 20th International Conference on Information Fusion (Fusion), Xi'an, 2017, pp. 1-8.

- The proposed RB^2-PF based visual odometry algorithm estimates the states of the camera and positions of the landmarks. Chapter 5 considers the estimated states (of camera

and landmarks) to improving the background modelling. While the camera is moving and video contains pronounced parallax, the proposed algorithm can significantly decrease the number of false alarms. The contribution was published in [17].
Y. Zhou and S. Maskell, "Moving object detection using background subtraction for a moving camera with pronounced parallax," 2017 Sensor Data Fusion: Trends, Solutions, Applications (SDF), Bonn, 2017, pp. 1-6.

- During the PhD, probabilistic hypothesis density (PHD) filter has been implemented and several applications has been discussed. Together with a gating method for data association, they partly contribute to [18].
C. Y. Liu, Y. Zhou, F. de Melo and S. Maskell, "Probabilistic graphical detector fusion for localization of faces and facial parts," 2014 Sensor Data Fusion: Trends, Solutions, Applications (SDF), Bonn, 2014, pp. 1-6.

1.3 Outline

The rest of this thesis is organised as follows.

Chapter 2 considers the moving object detection problem with a moving camera when the scene is almost flat. The problem consists of two tasks: global motion estimation (GME) and background subtraction. After describing the existing solutions of GME in Section 2.2, a robust cost function based on the student t-distribution is proposed. The proposed algorithm is derived from [19] and a novel parameter-tuning method is described. The advantage of the proposed algorithm is analysed and demonstrated in the context of a publicly available benchmark. With the GME results, a mosaic background model can be built using existing background modelling techniques (e.g., [20]). Section 2.4 shows the detection results on four datasets. Finally, a multi-target tracker (GM-PHD filter) is applied to the detections.

Chapter 3 reviews the problem of radar tracking and relevant solutions in Sections 3.2–3.4. In Section 3.4.4, an approach that combines the near-optimal proposal and Rao-Blackwellisation with correlated process noise is implemented. The idea is related to the work in [21]. A series of experiments is conducted to understand the utilities of the near-optimal proposal, Rao-Blackwellisation, and the combined approach in Section 3.4.5. The experiments consider several scenarios based on radar data with different noise uncertainties and dimensions.

Chapter 4 extends the work in Chapter 3 by using it to develop a solution to SLAM or visual odometry (VO) problems. The novel algorithm is called RB^2-PF based SLAM/VO. Starting with the two-dimensional SLAM problems in Section 4.2, it is shown that RB^2-PF outperforms the ordinary FastSLAM 2.0 method [22] when considering the near-constant velocity dynamic model. In Sections 4.3.1 and 4.3.2, the algorithm is then extended to working in three-dimensional space by introducing a pin-hole camera model and landmark representations from [23] and [24]. Monocular camera data are used as input, such that visual features [25] are considered for data association. Two public benchmarks are used to evaluate the proposed algorithm in Section 4.3.4.

Chapter 5 proposes a novel moving object detection algorithm for videos of non-flat scenes from a moving airborne camera. The videos contain pronounced parallax, which is significantly challenging for existing methods. In this chapter, a VO algorithm [16] is used to improve the background motion estimation. Then, areas of pronounced parallax are detected and used to refine the background model. Using these two steps facilitates removing most of the false alarms. An existing background subtraction algorithm [26] is used for detecting the moving objects. The details are described in Section 5.3.3. In Section 5.4, two videos of rural and urban areas are used for evaluating the proposed approach. It outperforms an existing algorithm on both datasets.

Chapter 6 concludes the thesis and identifies avenues for further research.

Chapter 2

Background Subtraction from a Moving Camera with a Flat Scene

2.1 Introduction

2.1.1 Moving Object Detection from a Moving Camera

Moving object detection is a significant part of surveillance systems. One of the common assumptions is that if nothing is moving, nothing is abnormal. This topic has been studied for several decades. One basic idea is to segment the object from the background using a background model. This approach has been popular in the context of stationary cameras for many years. However, surveillance cameras are sometimes not stationary. Indeed, pan-tilt-zoom cameras increase the field of regard by performing rotation and zooming (as considered in [27, 28]), while the position of the camera itself remains fixed. Similarly, the growing popularity of (e.g., unmanned) aerial vehicles provides a platform for collecting large-scale surveillance video but also introduces another three degrees of freedom (DoF) into the camera geometry. Both such camera systems motivate the development of more sophisticated background estimation techniques.

There are two broad classes of solutions to the problem of moving object detection from mobile camera systems. The first is to estimate the motion vectors of all the pixels. This class of solution is known as optical flow [29–31]. The interested reader is referred to various recent developments [32–34]. Optical flow based approaches do not construct the background explicitly. They are therefore not influenced by background changes or by the presence of pronounced parallax. However, their performance is heavily dependent on the quality of motion vectors extracted in the context of both foreground objects and the background. As a result, optical flow often works well only in the context of relatively large objects and a background that moves in a way that is globally consistent across the image. Optical flow also suffers from having a heavy computational cost while smoothing the velocity vectors of each pixel. Unfortunately, the scenarios motivating this chapter do not have relatively large objects and a globally consistent background. Of course, the aim is to develop techniques that minimise computational cost while maintaining algorithmic performance.

The second class of solution is based on extending traditional approaches to background subtraction and considering motion compensation. For example, [35] used the least median of squares method to estimate the background motion and assumed an affine transformation can align two consecutive frames. Moving objects could then be detected by subtracting the current image and the motion-compensated previous image. However, since this approach only compared two frames, it struggled to detect slowly moving objects and the objects that have recently stopped. Additionally, [36, 37] used an affine transformation to compensate for the background in the context of a moving camera. Moving objects were detected by subtracting the current image from a background estimate. This kind of approach assumes that objects and the background are all moving on a single (ground) plane. It is critical to the effective operation of such algorithms that motion compensation is tackled successfully. This motivates our research into global motion estimation (GME).

2.1.2 Global Motion Estimation

Global motion estimation aims to find the background motion between two images. It is widely used in video processing to infer camera movement (e.g., [38] [39]). The problem motivating this chapter is how to best extract background motion from a video that includes a significant number of pixels that are associated with moving foreground objects relative to those that are introduced by the recording device (e.g., salt and pepper noise and blurring caused by the camera being out of focus). The foreground pixels associated with moving objects are usually significant outliers and are near one another so that it is challenging to eliminate their influence on the estimated motion using standard image processing techniques (e.g., medium or Gaussian filters) [40]. The clusters of pixels associated with each foreground object also have consistent motions relative to the background. This chapter aims to solve the problem of GME in the context of such foreground pixels.

Global motion estimation (also can be called image registration) is not a new topic, and related research is extensive. In general, GME can be categorised into feature-based [41] [42], patch-based [43] [44], and pixel-based ('direct') [45] [46] approaches. The feature-based approaches match features based on the descriptors of features (e.g., Scale-Invariant Feature Transform (SIFT) [47], Speeded Up Robust Features (SURF) [48], Oriented FAST and Rotated BRIEF (ORB) [49], Binary Robust Invariant Scalable Keypoints (BRISK) [25], etc.) and then estimate the global motion given the matched features. Patch-based approaches use the raw pixel values to estimate the motion for the patches, assuming that the registration is constant over the entire patch, the global motion can be estimated by combining all the motions of the patches. Pixel-based approaches use the raw pixel values to estimate global motion while considering all pixels at one time. Several comparative studies have been conducted in the context of these approaches (see [46] [50]). These studies concluded that the pixel-based approach was the most accurate, especially in the context of video processing. The pixel-based approach is therefore the focus in this chapter.

Modern pixel-based approaches are based on the method developed by Lucas and Kanade [19]. There are numerous extensions that have been proposed over recent decades. For example, [30] is a well-cited review paper that described many of these extensions and listed their advantages. In the context of the original Lucas-Kanade approach, the L2 norm (also known as the quadratic cost function) was used to measure the error between two images. The algorithm exploited the fact that the L2 norm is both continuous and has a smooth first derivative [19]. This algorithm considered all the pixels during the optimisation process. It therefore offered some robustness to small quantities of outliers and some noise. However, several papers have identified that pronounced noise and outliers can degrade the estimation. Some techniques have been proposed to reduce the influence from outlier pixels while still adopting the L2 norm (e.g., [51]). There are also several papers that propose replacing the L2 norm by other robust cost functions that are commonly used in the statistical community. Examples include the L1 norm, Huber's M-estimator, Tukey's M-estimator, and the Cauchy-Lorentzian function. The details of these functions will be described in Section 2.3.1. These cost functions have all used 'soft' gates in the optimisation process (e.g., [51–55]). Indeed, these cost functions offer improved robustness to the presence of outliers by constraining or reducing the contribution to the cost function made by pixels associated with significant errors. However, they all have some disadvantages regarding either accuracy or computational efficiency. For example, using the L1 norm can be considered to be related to using the median estimator, which is less sensitive than a mean estimator (which is related to using the L2 norm). However, due to the discontinuous first differential of the L1 norm, minimising the L1 norm is more challenging than minimising the L2 norm. Although there are several ways to optimise the error function (e.g., as described in [53]), using an L1 norm still suffers from slower convergence and sensitivity response to poor initialisation. Huber's M-estimator limits the contribution to the cost function made by pixels with high errors, but the high-error pixels still have higher influence on Huber's M-estimator than on the other cost functions listed in Table 2.1. This limits the extent to which Huber's M-estimator is robust to the presence of outliers. Tukey's estimator adopts a more aggressive approach to limiting the contribution made by high-error pixels. However, some of the pixels that are currently considered to have high errors could be inliers with respect to the final estimate. This makes the optimisation process prone to falling into local minima. The Cauchy-Lorentzian function (sometimes known as the Cauchy function) offers good performance relative to the aforementioned functions. Note that it is known that the Cauchy-Lorentzian distribution is a special case of the student t-distribution. The corresponding curves are shown in Table 2.1.

The student t-distribution is a widely used robust function in the context of statistics (e.g., see in [56]). In the context of image registration, in the previous work, the student t-distribution had capitalised on the heavy-tailed and parameterised nature of the distribution. That work considered a complex optimisation of mixture of student t-distribution (e.g., [57] [58]). However, these methods did not consider working with video inputs that the computational cost is therefore significant. In this chapter, we will focus on processing videos. Our approach includes an online adaption of the parameters in response to a sequence of images such that the student t-distribution provides a cost function that interpolates between the L2 norm and the Cauchy-Lorentzian function.

2.2 Global Motion Estimation

This section will introduce affine and projective transformations for estimate motion between two images, and point out their utilities. Feature-based estimation and pixel-based estimation will be described and applications will be presented.

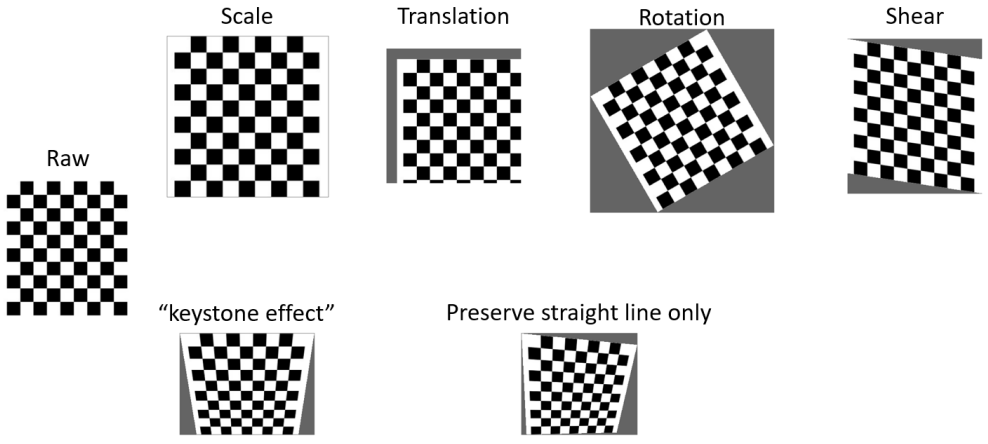
2.2.1 Problem Statement

Suppose we have two continuous frames from a video: I_1 and I_2. These two frames are taken from two different viewpoints. The task is to transform I_2, so it will align with I_1. According to multi-view geometry, it is possible to apply a transformation matrix H to each pixel with the equation:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = h\left(\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}, H \right), \tag{2.1}$$

where $\begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$ and $\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$ are the coordinates in images I_1 and I_2, respectively. Moreover, H is the transformation matrix, and $h(.)$ is the transformation function.

Several transformation functions exist. Two of them are usually used in GME: the affine transformation and the homography (projective) transformation. These two transformations will be briefly described and used in this chapter.



Affine transformation includes scale, translation, rotation and shear. Projective transformation includes all types of transformations in affine transformation and preserves straight lines, but parallelism can be changed.

FIGURE 2.1: A few examples of the affine transformation and the homography transformation

Affine transformations

The affine transformation (see examples in Figure 2.1) is a six DoF geometric transformation and its transformation function is defined as follows.

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = A \cdot \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \tag{2.2}$$

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix}. \tag{2.3}$$

The affine transformation can include the operations of rotation, translation, scale, and shear to an image. It can represent the image transformation from several kinds of camera motions. However, it cannot address the keystone effect, which is an apparent distortion when the scene is projected to another surface with different orientation.

The affine transformation is useful when the orientation of the camera relative to the scene plane is always similar. For example, supposing the camera is exactly looking down the ground, if it moves along x-axis, y-axis, z-axis and performs rotation along yaw-axis, but not performs rotation along pitch and roll-axis, the recorded images can be aligned to each other via affine transformation. Although in real applications an affine transformation is never an sufficient model, when the sampling rate is high, we can often use the affine transformation to approximate the true motion.

Homography transformations

The homography transformation is an eight DoF geometric transformation (see examples in Figure 2.1) and is described as follows:

$$\begin{bmatrix} u_1 \\ v_1 \\ w_1 \end{bmatrix} = H \cdot \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}, \tag{2.4}$$

$$H = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & 1 \end{bmatrix}, \tag{2.5}$$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} \frac{u_1}{w_1} \\ \frac{v_1}{w_1} \end{bmatrix}. \tag{2.6}$$

The biggest difference between the homography transform and the affine transform is that the homography transform does not preserve parallel lines. As a result, it can model the keystone effect. The homography transformation is therefore better able to model a perspective projection than the affine transformation. The homography transformation is

usually used in structure-from-motion approaches where the objective is to estimate the 3D structure of an object, as the camera should move around the object to get the appearances from multiple angles.

In this chapter, we consider an airborne camera that does not often change its attitude and has 25 Hz frame rate. The affine transformation is therefore used because it has fewer parameters and can be expected to typically exhibit faster convergence. Furthermore, as the affine transformation is six DoF, it requires fewer measurements to estimate the matrix. This is a significant consideration with a feature-based approach, but it is not a concern with a pixel-based approach.

2.2.2 Feature-based Global Motion Estimation

Estimating the transformation matrices should use corresponding points. Ways of extracting visual features will be introduced in this section. To avoid the influences from outlier corresponding features, robust methods will be described afterwards.

2.2.2.1 Transformation Matrix Estimation

The feature-based GME approach can be solved by direct linear transformation (DLT) and will be briefly described as follows. It is known that there are more than one solution. Considering the rotation and translation separately is one of them (e.g., [59]). However, it is hard to be extended to the homography transformation and thus, not to be described in this chapter.

The basic idea of DLT is to solve the following equations for the entries of affine transformation matrix, A, using Singular-value decomposition (SVD). Identical idea is widely used for estimating the homography (e.g., [60, 61]) in multiple view geometry.

$$\left\{ \begin{array}{c} c^{(1)}\mathrm{x}_1^{(1)} = A\mathrm{x}_2^{(1)} \\ c^{(2)}\mathrm{x}_1^{(2)} = A\mathrm{x}_2^{(2)} \\ \cdots \\ c^{(i)}x_1^{(i)} = Ax_2^{(i)} \end{array} \right\}. \tag{2.7}$$

where

$$\mathrm{x}_1^{(i)} = \left[x_1^{(i)}, y_1^{(i)}\right]^T, \tag{2.8}$$

$$\mathrm{x}_2^{(i)} = \left[x_2^{(i)}, y_2^{(i)}, 1\right]^T, \tag{2.9}$$

and (i) is the index of the matched points and $[c^{(1)}, ..., c^{(i)}]$ is a set of unknown non-zero scalars.

To get rid of the $c^{(i)}$ in (2.7), we introduce the anti-symmetric matrix:

$$\mathcal{H} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \tag{2.10}$$

Then, we can obtain the following equation by multiplying ${\mathrm{x}_1^{(i)}}^T \mathcal{H}$ from the left.

$$c^{(i)} {\mathrm{x}_1^{(i)}}^T \mathcal{H} \mathrm{x}_1^{(i)} = {\mathrm{x}_1^{(i)}}^T \mathcal{H} A \mathrm{x}_2^{(i)}. \tag{2.11}$$

Such that, the problem becomes solving the following equations:

$${\mathrm{x}_1^{(i)}}^T \mathcal{H} A \mathrm{x}_2^{(i)} = 0, (i = 1, 2, ...). \tag{2.12}$$

By substituting equations (2.3) (2.8) and (2.9) into Equation (2.12). It can be written as follows.

$$\mathcal{B}\mathcal{A} = 0, \tag{2.13}$$

where

$$\mathcal{B} = \begin{pmatrix} y_1^{(1)} x_2^{(1)} & y_1^{(1)} y_2^{(1)} & y_1^{(1)} & -x_1^{(1)} x_2^{(1)} & -x_1^{(1)} y_2^{(1)} & -x_1^{(1)} \\ y_1^{(2)} x_2^{(2)} & y_1^{(2)} y_2^{(2)} & y_1^{(2)} & -x_1^{(2)} x_2^{(2)} & -x_1^{(2)} y_2^{(2)} & -x_1^{(2)} \\ ... & ... & ... & ... & ... & ... \\ y_1^{(i)} x_2^{(i)} & y_1^{(i)} y_2^{(i)} & y_1^{(i)} & -x_1^{(i)} x_2^{(i)} & -x_1^{(i)} y_2^{(i)} & -x_1^{(i)} \end{pmatrix}, \tag{2.14}$$

$$\mathcal{A} = \begin{pmatrix} A_{11} \\ A_{12} \\ A_{13} \\ A_{21} \\ A_{22} \\ A_{23} \end{pmatrix}. \tag{2.15}$$

Finally, we perform Singular-value decomposition to matrix $\mathcal{B}$. If the matched points are exact without any noise, we should have a singular value that equals zero and the affine transformation matrix is identical to the corresponding right singular vector in the form of (2.15). If the

matched points are not precise, the singular values will not include zero. We therefore choose the right singular vector with the smallest singular value.

Extending to the homography transformation is straightforward. Except for re-writing (2.7) accordingly, we can choose any two from the following matrices, (2.16), to eliminate the constant $c^{(i)}$ and then perform a similar derivation.[1]

$$\mathcal{H}_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, \mathcal{H}_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \mathcal{H}_1 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{2.16}$$

Note that both the estimated affine and homography transformation matrices may not be scaled to the input images, as we defined the scalars $c^{(i)}$ at the beginning. In practice, we can assume the same scale for all the matched points. The simplest way to find out such constant is to perform the unscaled transformation to all the matched points and then estimate the scale. For homography transformation, we can otherwise simply find out the scale that normalises H_{33} to 1.

2.2.2.2 Image Feature Descriptors

The key to estimating accurate transformation matrices is to get enough number of precise matched points. The matched points can either be generated from a feature tracker or a feature matching method. The feature points are called 'keypoints' in related papers and the name will be used in this chapter.

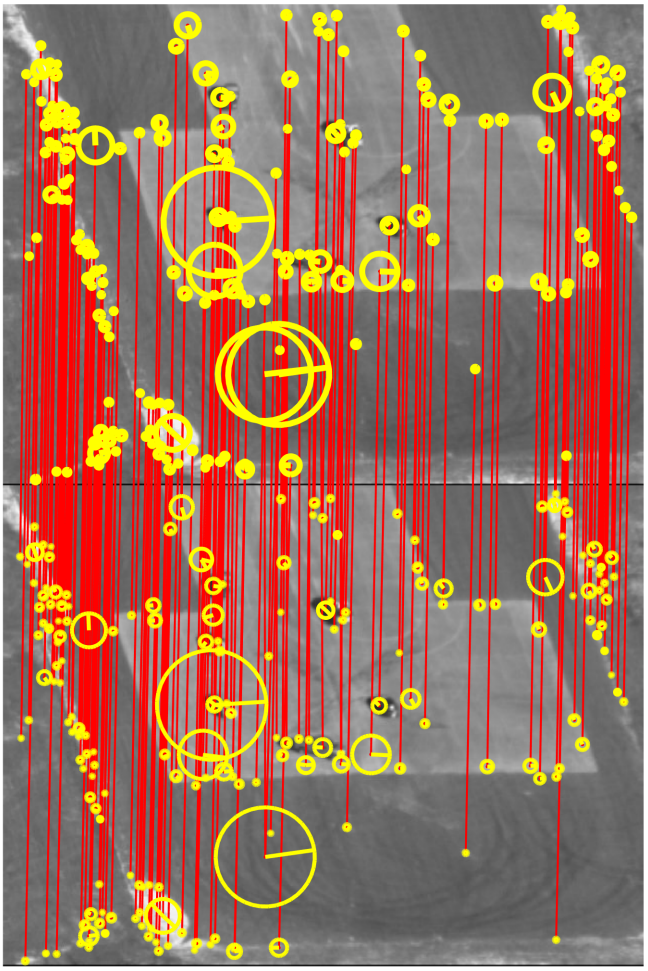
A tracker-based approach (e.g., the Kanade–Lucas–Tomasi (KLT) tracker [62]) usually uses a corner detector (e.g., FAST [63, 64] or AGAST [65]) to detect the keypoints. Then it tracks those keypoints according to the image gradients via optical flow [19] to get matched points. The advantage of this approach is that it is quick and simple. However, it is critically reliant upon the distribution of the keypoints and the accuracy of the tracks. This approach does not work when the displacement between two frames is large which is a known drawback of optical flow.

A matching-based approach calculates features (e.g., SIFT [47], SURF [48], ORB [49] and BRISK [25]) for particular points in both images. The matching is conducted by only using the value of feature descriptor without considering the spatial information. Such that it overcomes the problem that the features are difficult to track while the image displacement is large. Both SIFT and SURF proposed their own ways to extract the keypoints. SIFT calculates the Laplacian-of-Gaussian (LoG) of a image in different scales to identify the scale-space extrema.

[1] There are enormous tutorials discussing how to estimate the homography transformation matrix given matched points. Those methods are equivalent to choosing $\mathcal{H}_1$ and $\mathcal{H}_2$, though most of them did not explicitly introduce such matrices.

SURF improves this by using Differences of Boxes by integral image [66] which is more computational efficient. After obtaining the keypoints, SIFT computes the gradient of a region centred by the feature point and SURF computes the Haar Wavelet. For more detail of these two methods, refer to [47] and [48]. ORB and BRISK are binary descriptors, such that hamming distance is used to find the most similar features. They are proposed to be fast and claimed to be hundreds times faster than SIFT, but the accuracy of matching is reasonably lower. Ways to avoid outliers will be described in Section 2.2.2.3. The basic idea of this kind of binary descriptors is to compare the intensity values between two pixels that is on a pattern of sampling. BRISK uses a fixed sampling pattern, while ORB uses a learned sampling pattern by Principle Component Analysis. There is no independent feature point detector proposed from these two methods, and any corner detector (e.g., FAST, AGAST, etc.) can be used. We refer to [49] and [25] for more descriptions. Moreover, for GME, it also relies on the distribution of features.

We show an example of feature matching in Figure 2.2 where SIFT features are used.



The feature points are generated by SIFT. The scale and the outstanding orientation of the feature descriptor is shown by the yellow circle and the yellow line. The red line shows which two feature points are matched.

FIGURE 2.2: An example of matched features between two frames.

2.2.2.3 Random Sample Consensus

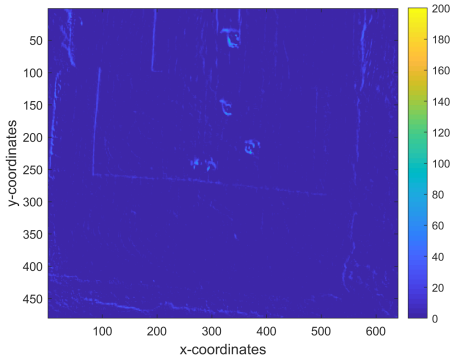
Since the matching is prone to error and the existence of outliers is inevitable, to obtain an accurate estimation of the parameters, we usually need more matched points than the minimum to observe the parameters. However, least square is used to calculate the transformation matrix and it has relative weak robustness. The obvious outliers (e.g., the matched points on the moving vehicles in Figure 2.2) can therefore cause large error. Random sample consensus (RANSAC) [67] is proposed to solve this kind of problems. A brief processing chain of GME with RANSAC is described in Algorithm 1.

Algorithm 1 Pseudo-code for global motion estimation (affine) with random sample consensus.

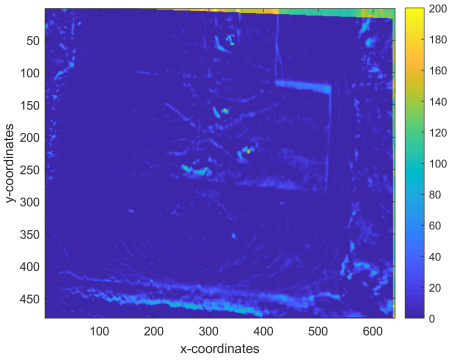
1: Suppose we have a set of matched points $S_1^{all} = \{s_1^{(i)}, ...\}$ and $S_2^{all} = \{s_2^{(i)}, ...\}$ from I_1 and I_2 respectively, and configure a maximum iteration number N_{iter} and a threshold t for accepting the inliers.
2: Initialise the parameters as follows. The inliers sets $S_1^{inlier} = \emptyset$ and $S_2^{inlier} = \emptyset$, and the current iteration number $C_{iter} = 0$
3: **while** $C_{iter} < N_{iter}$ **do**
4: Pick three matched points randomly from S_1^{all} and S_2^{all}.
5: Calculate the affine transformation matrix, A.
6: Apply the affine transformation A to all the points in S_1^{all} and get $\check{S}_1^{all}$.
7: **for** each point, $\check{s}_1^{(i)}$, in $\check{S}_1^{all}$ **do**
8: Calculate the distance $D = distance\left(\check{s}_1^{(i)}, s_2^{(i)}\right)$.
9: **if** $D < t$ **then**
10: Put $s_1^{(i)}$ into S_1^{inlier}.
11: Put $s_2^{(i)}$ into S_2^{inlier}.
12: **end if**
13: **end for**
14: **if** $|S_1^{inlier}| < |S_{test}|$ **then**
15: $S_1^{inlier} = S_1^{test}$
16: $S_2^{inlier} = S_2^{test}$
17: **end if**
18: $C_{iter} = C_{iter} + 1$
19: **end while**
20: Calculate the affine transformation matrix A_{final} using $S_1^{inliers}$ and $S_2^{inliers}$.

RANSAC uses a hard parameter, t to accept inliers, therefore the choice of the threshold takes a significant role. It is known that if we want the estimation to be precise, t should be small. However, if the input data is noisy, there can be few inliers which makes the estimation less reliable. On the contrary, if using a large t when the matching is robust, some outliers will degrade the estimation as well. The solution starts with building the error (noisy feature descriptor and mis-matching) which can be a mixture of Gaussian and uniform distribution. [68] described to use a probability distribution to reject outliers and the contribution of the inliers are weighted as well. The objective is then finding out the set of weight inliers that can provide the maximum likelihood. [68] made a further improvement that Expectation Maximisation (EM) can be considered to estimate γ, which is defined as the parameter that is related to the aforementioned mixture distribution. This approach is therefore called Maximum Likelihood Estimation Sample Consensus (MLESAC). The details can be found in [68].

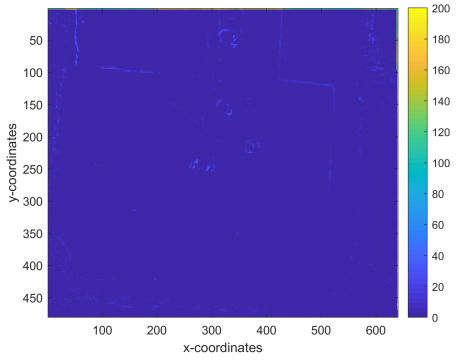
The results of estimating the global motion using feature-based solution using SIFT descriptors without robust techniques, with RANSAC or MLESAC are shown in Figure 2.3. In order to show the significance of using different robust techniques, Gaussian ($\mu = 0$ and $\sigma = 0.1$) noise is added to both frames before the GME process. For the visualisations, only the original frames are involved, thus the Gaussian noise does not influence the displayed errors. From the figures, using a robust technique is a necessity: the estimated transformation is obviously faulty with only considering the matched points. Both RANSAC and MLESAC produce good results, but MLESAC is better with a smaller total error. Moreover, RANSAC provides rather different estimations while t changes, this examples shows the result when $t = 1$ which is the best after a number of exhaustive searches.



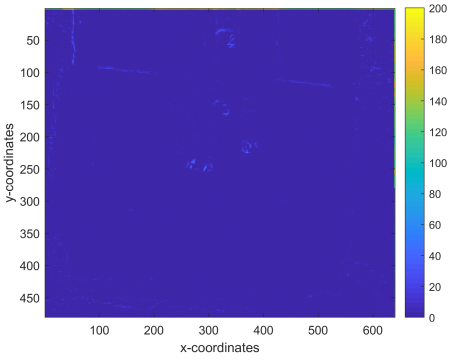
(a) The error (106135) between two original frames.



(b) The error (193392) between the first frame and the registered second image using the feature-based approach.



(c) The error (65222) between the first frame and the registered second frame using the feature-based approach and RANSAC.



(d) The error (59037) between the first frame and the registered second frame using the feature-based approach and MLESAC.

This example shows motion estimation result between frame 360 and 361 in the video EgTest01 of Vivid dataset. The value of the error (in the brackets) is represented by the extended L0 norm (see (2.47)). The scale of the error is represented by the colour bar on the right hand side.

FIGURE 2.3: An example of motion estimation via feature-based approach with either RANSAC or MLESAC.

2.2.3 Pixel-based Global Motion Estimation

Pixel-based GME considers all the pixels in the image. It is more precise and robust than feature-based approaches, especially with low quality images (e.g., caused by low light levels or out-of-focus images). Intuitively, the computational cost can be large. However, when the displacement between two frames is small, such as when considering a video with a 25 Hz frame rate and when the resolution of the frame images is acceptably low (e.g., 640×480 in [69]), it can take a reasonable time to converge. Another technique that is usually used to deal with large displacement between two frames is the image pyramid (the coarse-to-fine method) [70]. The images are scaled, usually by a factor of 2 (i.e., quadtree). The optimisation starts from the smallest scale, and the motion estimation variables are propagated to the larger scale. This technique accelerates the GME process and can be useful to avoid local minima.

Pixel-based GME has been proposed and widely used for decades, so it will only be briefly described here. Note that we only consider the affine transformation in this chapter.

The core of pixel-based GME is to solve an optimisation problem, finding a transformation matrix, H, to minimise $\varepsilon(.)$ as follows:

$$\varepsilon(H) = \sum_{x=1}^{\omega_x} \sum_{y=1}^{\omega_y} \rho \left(I_1 \begin{pmatrix} \mathrm{x} \\ \mathrm{y} \end{pmatrix} - I_2 \left(H \cdot \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ 1 \end{bmatrix} \right) \right), \tag{2.17}$$

where $[x, y]^T$ is the coordinate on the image I_1, ω_x and ω_y are the sizes of the image on the x-axis and y-axis, $I_1(.)$ and $I_2(.)$ represent pixels on image I_1 and I_2 respectively, given the coordinates. $\rho(.)$ is the cost function.

As previously explained, the choice of cost function can have a significant effect on both efficiency and accuracy. An ordinary pixel-based GME algorithm (as in [71]) uses the quadratic (the L2 norm) cost function in (2.18). Since the cost is quadratic, the convergence can be faster than some other robust cost functions. However, using an L2 norm means that the optimisation is not robust, such that the algorithm can be adversely influenced by outliers. The cost function is as follows:

$$\rho(x) = x^2. \tag{2.18}$$

After the cost function is chosen, there are several optimisation methods that can be used. The GME problem is challenging since the images are often heavily textured, which can lead to the existence of local minima. Fortunately, for video processing, we assume that the displacement between the two images is small. Therefore, the Newton's method is commonly used and will be implemented in the subsequent experiments. The inference and pseudo-code are already well documented in many existing papers (e.g., [19] and [71]), so they will not be displayed in this

chapter.

Figure 2.6 shows the results after the pixel-based GME approach as a comparison without using a contribution mask [2]. It uses the same input frames as Figure 2.3, and some differences are evident when inspecting the edge of the runway.

2.3 Robust Global Motion Estimation

2.3.1 Cost Functions

In this subsection, a few existing and widely used robust cost functions will be described. The $\rho(x)$ and the $\psi(x)$ curves are displayed in Table 2.1. The $\rho(x)$ curve is the cost function itself, and the $\psi(x)$ curve is the differentiation of the cost function. These two curves are introduced while describing the M-estimators [72], but they are useful to visualise the property of all the cost functions. $\psi(x)$ is also called an 'influence function', which is a significant measurement of the robustness. It will be used for analysing the robustness of each cost function in the following subsections.

2.3.1.1 L1 Norm

The L1 norm is a well-known robust cost function for which the scale of errors has a minimal effect, which is described in (2.19). Its derivative is described in (2.20). Compared to the L2 norm, it is usually treated as a median estimator and is less sensitive to large errors. However, due to the discontinuous first derivative, the minimisation with respect to the L1 norm is computationally challenging. There are papers that provide sophisticated approaches to address the problems resulting from the discontinuities in the first derivative (e.g., [50] and [73]). However, we choose not to consider such methods extensively here. The $\rho(x)$ and the $\psi(x)$ of the L1 norm are defined as follows.

$$\rho(x) = |x|. \tag{2.19}$$

$$\psi(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases} \tag{2.20}$$

[2] The boundary area of one image can be transformed to positions that are outside the other image. If this was not carefully considered, this issue would give rise to numerous outliers. Hence, a contribution mask that prevents the pixels in the boundary of an image from contributing to the motion estimation, is created. The details will be investigated in Section 2.3.5.1.

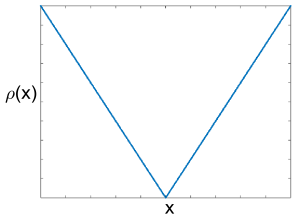
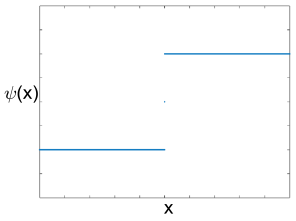
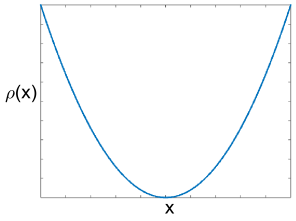
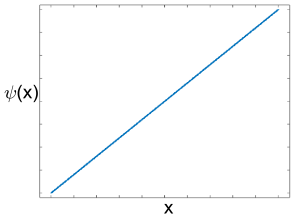
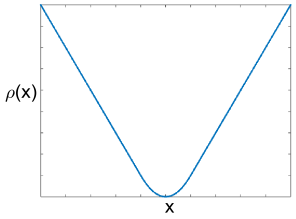
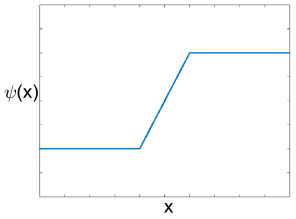
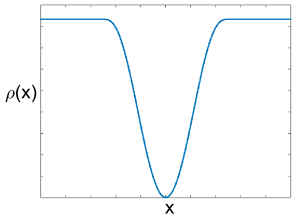
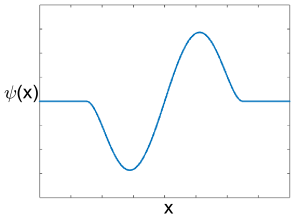
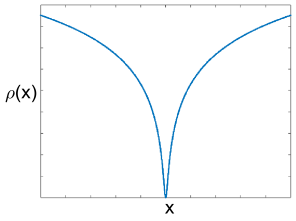
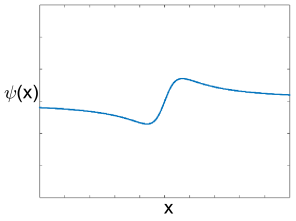
Techniques	Cost Function Curve, $\rho(x)$	Differentiation of the Cost Function, $\psi(x)$
L1 Norm		
L2 Norm		
Huber's M-estimator		
Tukey's M-estimator		
Cauchy-Lorentzian		

TABLE 2.1: Curves of existing cost functions.

2.3.1.2 M-estimators

The M-estimators are motivated by robust statistics, and they are designed to try to reduce the effects of outliers. Both least squares and maximum-likelihood estimation are known as popular M-estimators. The popular definition of these was proposed by Huber in 1964 as follows:

$$\hat{\theta} = \arg\min_{\theta} \left(\sum_{i=1}^{n} \rho(x_i, \theta) \right), \tag{2.21}$$

where $\hat{\theta}$ is the minimisation of the estimation $\sum_{i=1}^{n} \rho(x_i, \theta)$, ρ is a function with certain properties, which will be described later, and x_i denotes the observations.

To find the minimisation, (2.21), it can be helpful to find the derivative of $\rho(.)$ with respect to θ. If $\rho(.)$ is differentiable, the M-estimator will be called Ψ-type . Such M-estimators are discussed in this chapter, and they are also widely used in GME. In the context of an M-estimator, the minimisation problem becomes solving equation (2.22).

$$\int_{\chi} \psi(x, \theta) dF(x) = 0 \tag{2.22}$$

where $\psi : \chi \times \Theta \rightarrow R^r$ maps a probability distribution F on χ. For the situation in (2.21), $\psi(x) = \frac{\partial \rho(x)}{\partial x}$.

Huber's M-estimator

The ρ and ψ functions of Huber's M-estimator [74] are given in (2.23) and (2.24). The influence of x increases linearly when $|x| \leq c$, and then remains unchanged when $|x| > c$, where c is an adjustable parameter (as shown in Table 2.1). For GME, this means the influence from lower-error pixels has the same influence as that in the L2 norm, and the influence from higher-error pixels is constrained to some fixed value. Changing c can control the influence from high-error pixels, but the slope cannot be changed. The influence of outliers is limited, but they still have a greater influence on the cost function than any low-error pixels. The $\rho(x)$ and the $\psi(x)$ of Huber's M-estimator are shown as follows.

$$\rho(x) = \begin{cases} \frac{x^2}{2} & if\ |x| < c \\ c(|x| - \frac{k}{2}) & if\ |x| \geq c \end{cases} \tag{2.23}$$

$$\psi(x) = \begin{cases} x & if\ |x| < c \\ c \cdot sgn(x) & if\ |x| \geq c \end{cases} \tag{2.24}$$

Tukey's M-estimator

The ρ and ψ functions for Tukey's M-estimator [75] are presented in (2.25) and (2.26). Inspecting the graphs of $\rho(x)$ and $\psi(x)$ in Table 2.1, the influence of x allows outliers to have less of an influence than in the context of Huber's M-estimator. The influence of x increases as x moves away from zero and then decreases when $|x| \leq c$. It drops to 0 when $|x| > c$. Moreover, c is an adjustable constant. Therefore, the influences from the outliers are suppressed more than with Huber's M-estimator. For GME, this means that the influence from lower-error pixels is more significant than for the L2 norm. However, the higher-error pixels will not influence the convergence, which is robust but could be a drawback under some circumstances. The effect of changing k is similar to Huber's M-estimator, and the slope changes along with the value of x.

$$\rho(x) = \begin{cases} \frac{c^2}{6}\left(1 - [1 - (\frac{x}{c})^2]^3\right) & if\ |x| < c \\ \frac{c^2}{6} & if\ |x| \geq c \end{cases} \tag{2.25}$$

$$\psi(x) = \begin{cases} x\left(1 - (\frac{x}{c})^2\right)^2 & if\ |x| < c \\ 0 & if\ |x| \geq c \end{cases} \tag{2.26}$$

2.3.1.3 Cauchy-Lorentzian Function

The Cauchy-Lorentzian function is derived[3] from the Cauchy-Lorentzian distribution. They are defined in (2.27) and (2.28). Its curve looks like the Gaussian distribution near zero, but it can be more robust than the Gaussian distribution because it has a heavy tail. The $\rho(x)$ and $\psi(x)$ of the Cauchy-Lorentzian function are shown in (2.27) and (2.28). The influence curve of the Cauchy-Lorentzian function shares properties with Huber's M-estimator and Tukey's M-estimator. The influence increases when $|x| < c$ and decreases when $|x| > c$. Note that, unlike Huber's and Tukey's M-estimators, c is a implicit value that is related to but distinct from σ. Nonetheless, the influence does not decrease to 0 in the same way that Tukey's M-estimator does. The curves of ρ and ψ are shown in Table 2.1.

$$\rho(x) = log\left(1 + \frac{1}{2}\left(\frac{x}{\sigma}\right)^2\right) \tag{2.27}$$

$$\psi(x) = \frac{2x}{2\sigma^2 + x^2} \tag{2.28}$$

By inspecting the curves, it can be observed that the Cauchy-Lorentzian function is less influenced by outliers than Tukey's M-estimator. This could be an advantage when considering GME. Not all the high-error pixels are caused by outliers, and some can just be the result of a background that is rich in texture. If those pixels make no contribution to the cost function as the algorithm converges, it may increase the time taken to converge or, worse, lead to the optimisation becoming stuck in the local minima. The shape of the ψ curve shows that it is

[3]The function is the natural logarithm of the likelihood such that the function is the log-likelihood.

likely to be better than any of the other cost functions described up to this point for robust GME. Nevertheless, there is only one parameter, σ, of the Cauchy-Lorentzian function, and this parameter controls the magnitude of the curve. That is, the value for c cannot be explicitly adjusted. For GME, this means that the boundary between high errors and low errors cannot be configured explicitly by modifying a parameter.

2.3.2 Student t-distribution

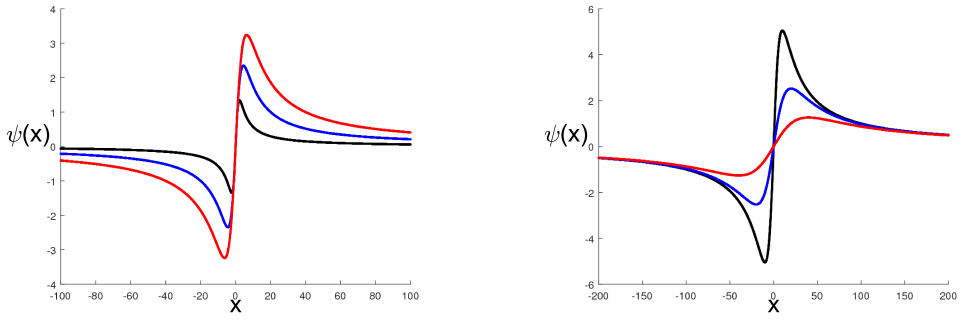


FIGURE 2.4: The influence function of the traditional parameterisation of the student-t function in (2.31). (left) $\sigma = 1$, $v = 5, 20, 40$; (right) $\sigma = 1, 2, 4$, $v = 100$.

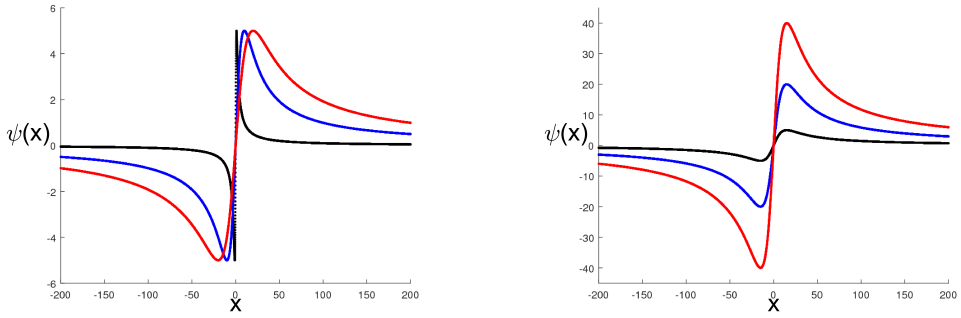


FIGURE 2.5: The influence function of the re-parameterised student-t function in 2.33. (left) $\tau = 5$, $\nu = 1, 10, 20$; (right) $\tau = 5, 20, 40$, $\nu = 15$.

How to choose a robust cost function that will facilitate efficient optimisation is not obvious. Due to its ability to interpolate between the Gaussian distribution (when $v = \infty$) and the Cauchy-Lorentzian distribution (when $v = 1$), the student t-distribution (in (2.29)) can be used in a way that adapts in response to each of a series of optimisation problems. In order to use it as a cost function, the student t-distribution (2.29) is expressed in (2.30) on a logarithmic scale. It is a common way to obtain a error function (e.g., [76]). Note that we also re-parameterise the distribution in a way that makes the interpolation explicit between the L2 norm and the Cauchy-Lorentzian function. More specifically, this interpolation is parameterised by v. The differentiation of the student-t function is (2.31).

$$P(x) = \frac{\Gamma(\frac{v+1}{2})}{\sqrt{v\pi\sigma^2}\Gamma(\frac{v}{2})}\left(1+\frac{x^2}{v\sigma^2}\right)^{-\frac{v+1}{2}}, \tag{2.29}$$

$$\rho(x) = \frac{v+1}{2}\ln\left(1+\frac{x^2}{v\sigma^2}\right) - \ln\left[\frac{\Gamma(\frac{v+1}{2})}{\sqrt{v\pi}\sigma\Gamma(\frac{v}{2})}\right], \tag{2.30}$$

$$\psi(x) = \frac{(v+1)x}{v\sigma^2+x^2}. \tag{2.31}$$

Figure 2.4 shows how the influence functions change when only v or only σ changes. By considering the peak on the curve, it is obvious that neither v nor σ alone can be used to determine the peak or its magnitude. When either v or σ increases, the peak becomes larger and moves away from the origin. In the context of GME, this could be confusing since it would be difficult to decide from which value (i.e., pixel error) the influence curve begins to decay or what the bound of the magnitude of the influence would be. Therefore, we propose the parameterisation of the student-t cost function in (2.32), and its influence function is in (2.33):

$$\rho(x) = \tau\nu\ln\left(1+\frac{x^2}{\nu^2}\right) - \ln\left[\frac{\Gamma(\tau\nu)}{\sqrt{\pi}\nu\Gamma(\tau\nu-\frac{1}{2})}\right], \tag{2.32}$$

$$\psi(x) = \frac{2\tau\nu x}{\nu^2+x^2}. \tag{2.33}$$

The re-parameterisation replaces v and σ by ν and τ. Note that this incurs no additional computational cost. Figure 2.5 shows how the cost function changes when modifying ν and τ. It should be clear to the reader that the position (on the x-axis) of the peak point equals ν and the magnitude of the peak point (on the y-axis) equals τ. The re-parameterisation provides a clear interpretation to both parameters. In the context of optimisation, a larger ν means that high-error samples have a greater influence. Such a larger ν results in an estimator that is less robust. However, since more samples are involved in the calculation, especially since the larger errors are included, the optimisation should be quicker and more robust to poor initialisation. Thus, the optimal compromise between efficiency, response to poor initialisation, and robustness becomes difficult to specify. The other parameter τ decides the maximum influence of a datum. This parameter behaves like a learning rate in an iterative algorithm. Thus, τ will not be our focus. In this chapter, it is recommended to be the same as ν, which makes the curve similar to the L2 norm when the absolute error is not bounded by ν.

2.3.3 Optimisation

Global motion estimation is to find a set of motion parameters that can be used to align the input image, I_{input}, according to the reference image. As mentioned, considering the altitude

of the airborne camera, affine transformation is discussed in this chapter. Recall the affine transformation model as in (2.34):

$$\begin{bmatrix} x & y \end{bmatrix}^T = f(x', y'; A) = A \cdot \begin{bmatrix} x' & y' & 1 \end{bmatrix}^T, \tag{2.34}$$

where $A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix}$ is the transformation matrix containing motion parameters to be calculated. Moreover, $\begin{bmatrix} x & y \end{bmatrix}$ is the warped coordinator, and $\begin{bmatrix} x' & y' \end{bmatrix}$ is the original coordinator.

The GME problem can be formulated as (2.35) with the student-t cost function. The total error ε should be minimised by estimating A:

$$\varepsilon = \sum_{x=1}^{\omega_x} \sum_{y=1}^{\omega_y} \tau\nu \ln \left(1 + \frac{\left(I_{ref}^{(x,y)} - I_{reg}^{(x,y)} \right)^2}{\nu^2} \right) - \ln \left[\frac{\Gamma(\tau\nu)}{\sqrt{\pi}\nu\Gamma(\tau\nu - \frac{1}{2})} \right], \tag{2.35}$$

$$I_{warp} = F(I_{input}, A), \tag{2.36}$$

where $F(.)$ is the warping process, and image interpolation may be applied during the warping if necessary. Further, I_{warp} is the warped image.

The Newton's method is used to solve the optimisation problem, and the first-order derivative is (2.37):

$$\frac{\partial \varepsilon}{\partial A} = \sum_{x=1}^{\omega_x} \sum_{y=1}^{\omega_y} \frac{-2\tau\nu \left(I_{ref}^{(x,y)} - I_{warp}^{(x,y)} \right)}{\nu^2 + \left(I_{ref}^{(x,y)} - I_{warp}^{(x,y)} \right)^2} \cdot \vartheta(x, y), \tag{2.37}$$

$$\vartheta(x, y) = \frac{\partial I_{warp}^{(x,y)}}{\partial A}, \tag{2.38}$$

where τ and ν are the parameters in the student-t function.

The second-order derivative of ε is as in (2.39):

$$\frac{\partial^2 \varepsilon}{\partial A^2} = \sum_{x=1}^{\omega_x} \sum_{y=1}^{\omega_y} \left(\frac{2\tau\nu(\nu^2 - I_{diff}^2)}{\nu^2 + I_{diff}^2} \cdot \vartheta(x, y)^T \cdot \vartheta(x, y) + \frac{2\tau\nu I_{diff}}{\nu^2 + I_{diff}^2} \cdot \frac{\partial}{\partial A'} \vartheta(x, y) \right), \tag{2.39}$$

$$I_{diff} = I_{ref}^{(x,y)} - I_{warp}^{(x,y)}. \tag{2.40}$$

We eliminate the second derivative term for reducing the computational cost. Thus, (2.41) is used for minimisation:

$$\frac{\partial^2 \varepsilon}{\partial A^2} = \sum_{x=1}^{\omega_x} \sum_{y=1}^{\omega_y} \left(\frac{2\tau\nu(\nu^2 - I_{diff}^2)}{\nu^2 + I_{diff}^2} \cdot \vartheta(x,y)^T \cdot \vartheta(x,y) \right). \tag{2.41}$$

To calculate ϑ, we consider equation (2.42):.

$$\vartheta(x,y) = \frac{\partial I_{warp}^{(x,y)}}{\partial \begin{bmatrix} x \\ y \end{bmatrix}} \cdot \frac{\partial f(x',y';A)}{\partial A} = \nabla I_{warp}^{(x,y)} \cdot \frac{\partial f(x,y;A)}{\partial A}, \tag{2.42}$$

where $\nabla I_{warp}^{(x,y)}$ is the image gradient $[\frac{\partial I_{warp}^{(x,y)}}{\partial x}, \frac{\partial I_{warp}^{(x,y)}}{\partial y}]^T$ relative to the x-axis and y-axis; $\frac{\partial f(x',y';A)}{\partial A} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}$. is from (2.34).

Therefore, (2.43) is used in the optimisation:

$$\vartheta = [\ x\nabla I_{warp}^x \quad y\nabla I_{warp}^x \quad \nabla I_{warp}^x \quad x\nabla I_{warp}^y \quad y\nabla I_{warp}^y \quad \nabla I_{warp}^y\ ]. \tag{2.43}$$

According to the Newton's method, the transformation matrix A was updated iteratively via (2.44)-(2.45) and initialised with (2.46). Moreover, a learning rate λ is mentioned, which is usually 1. In practice , we may need to use a smaller learning rate when the reference image and input image are highly inconsistent to prevent a wrong estimation. In [71], to reduce the computational cost, ∇I_{ref} can be used to alter ∇I_{warp}, which is only calculated once before the iterative estimation. Note that the coarse-to-fine technique (introduced in Section 2.2.3), which is usually used in image registration, is also implemented.

$$\nabla A = \lambda \cdot \left(\frac{\partial^2 \varepsilon}{\partial A^2} \right)^{-1} \cdot \frac{\partial \varepsilon}{\partial A} \tag{2.44}$$

$$A_{est} = \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} + \nabla A \right) \cdot A_{pre} \tag{2.45}$$

where A_{pre} is the estimated transformation matrix in previous iteration and A_{est} is the estimated transformation matrix in current iteration.

$$A_{init} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.46}$$

2.3.4 Parameter Estimation for the Cost Function

The involvement of τ and ν is an advantage of the proposed parameterised student-t function, and its value is significant. Note that, in Newton's method, τ is eliminated (see (2.37),(2.39)and(2.44)). In order to find the best ν that trades off the accuracy and time cost, a binary search-like scenario is adopted (see Algorithm 2). Since the images in a sequence tend to have similar content in a small period, ν is unnecessary to estimate often. Algorithm 2 is only activated when the global error, $\sum_{x=1}^{\omega_x}\sum_{y=1}^{\omega_y}(I_{ref}^{(x,y)} - I_{reg}^{(x,y)})$, changes considerably compared to previous estimations.

Algorithm 2 Parameter estimation scenario.

1: Perform global motion estimation with two potential parameters ν_{max} and ν_{min}. The errors and iteration numbers for optimisation are recorded. This step could be done using a small image.
2: **if** two errors are not close (measuring by some threshold). **then**
3: Keep ν (ν_{max} or ν_{min}) with smaller error. The other one will be replaced by $\frac{\nu_{max}+\nu_{min}}{2}$. Do Step 1.
4: **end if**
5: Choose the ν with the smaller iteration number for optimisation.

2.3.5 Experiments of Global Motion Estimation

We compare the performances using four datasets (EgTest01 - EgTest04) in the Vivid benchmark[4] [69]. The image resolution is 640×480. The images contain no occlusion, but some noise and sometimes blurring. The camera motion can be modelled by the affine transformation. To find the effects of poor initialisation to different cost functions, the sampling rate is reduced. In terms of evaluation, since the mean square error is easily dominated by outliers, we used an extended L0 norm, (2.47):

$$E = \sum_{x=1}^{P_x}\sum_{y=1}^{P_y}\varepsilon_{x,y}, \tag{2.47}$$

$$\varepsilon_{x,y} = \begin{cases} 1 & if|I_1^{(x,y)} - I_2^{(x,y)}| > c \\ 0 & if|I_1^{(x,y)} - I_2^{(x,y)}| \leq c \end{cases}$$

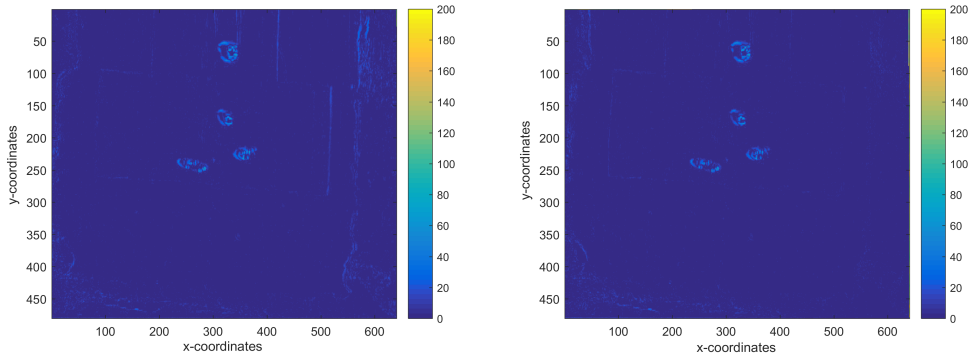
where $I_1^{(x,y)}$ and $I_2^{(x,y)}$ are pixels on two images and c is a constant. When $c = 0$, clearly it is the L0 norm. Further, $c = 2$ in all the experiments.

[4] http://vision.cse.psu.edu/data/vividEval/datasets/datasets.html

The experiment results will be reported in two parts. 1) The visualised estimation results will be displayed with picked pairs of consecutive frames. The accuracy between the efficient cost function (quadratic cost) and the proposed parameterised student-t cost function (Section 2.3.2) will be analysed. 2) The statistics on the entire datasets using all the mentioned cost functions will be presented. The improvements on both accuracy and efficiency will be described.

2.3.5.1 Results of Single Estimation

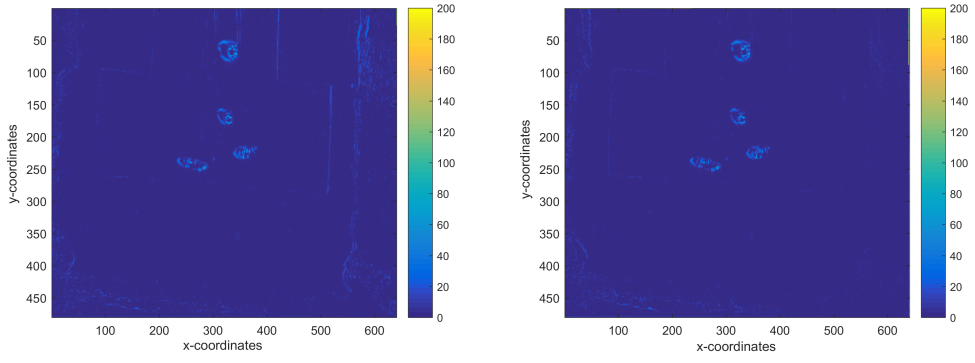
Global Motion Estimation without Contribution Mask



(a) Using quadratic cost
(Error (extended L0 norm, c=2): 75601)

(b) Using student-t cost
(Error (extended L0 norm, c=2): 48137)

FIGURE 2.6: Comparing the image registration error between using quadratic cost and student-t cost on Frames 360-361. The contribution mask is not applied.



(a) Using quadratic cost
(Error (extended L0 norm, c=2): 109467)

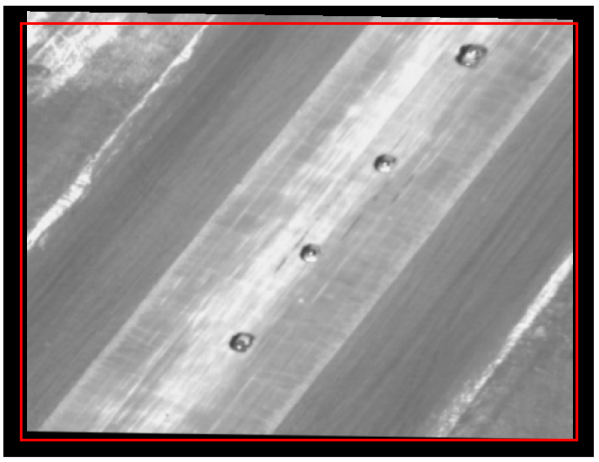
(b) Using student-t cost
(Error (extended L0 norm, c=2): 73420)

FIGURE 2.7: Comparing the image registration error between using quadratic cost and student-t cost on Frames 650-651. The contribution mask is not applied.

Frames 360-361 and Frames 650-651 are picked for testing the pixel-based GME using quadratic cost function (described in Section 2.2.3) and the pixel-based GME using student-t cost function with auto-tuning parameters (described in Section 2.3.2). Figures 2.6 and 2.7 visualise the registration error of each pixel. From these two figures, the improvement from

the student-t cost is evident. However, by inspecting the metric, we noticed that, for Frames 650-651, the quadratic cost based algorithm did not estimate the transformation matrix successfully. The extended L0 norm error is close to the error when the transformation is not applied. Furthermore, the student-t cost based approach does not provide a satisfying transformation estimation. We can still observe highlighted edges in the figures apart from the moving cars. We investigated the causes as follows.

The pixel-based approach considers all the pixels during estimating the global motion iteratively. Recall that the objective is to warp the input image, I_{input}, to get I_{warp}, so that the final I_{warp} aligns with I_{ref}. After one iteration of GME, we calculate I_{warp}. It is usually that there are some pixels in I_{warp} that cannot be sourced from I_{input}. Those pixels are usually filled by a fixed value (see Figure 2.8 for example when the pixels are filled by "black pixels"). For the next iterations, we calculate the GME between I_{warp} and I_{ref}. Therefore, the filled pixels distract the optimisation and so on in the subsequent iterations. It can be imagined that this issue is more obvious when the displacement between I_{input} and I_{ref} is larger. Unfortunately, before every iteration, we cannot identify what pixels will take part in the optimisation. A simple idea is to apply a contribution mask in the middle of the image.



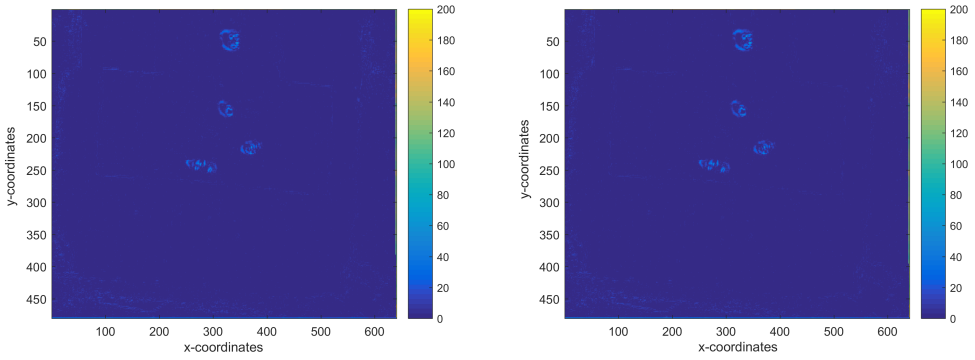
An example of I_{warp} is presented. Red box is the position of I_{ref} so the inner pixels contribute to the GME in each iteration. Some pixels are filled by black values, as I_{input} does not have that information. Without contribution mask, the error between I_{warp} and I_{ref} includes the filled pixels that suppose to be irrelevant.

FIGURE 2.8: This image shows a warped frame and the contributed pixels.

Global Motion Estimation with Contribution Mask

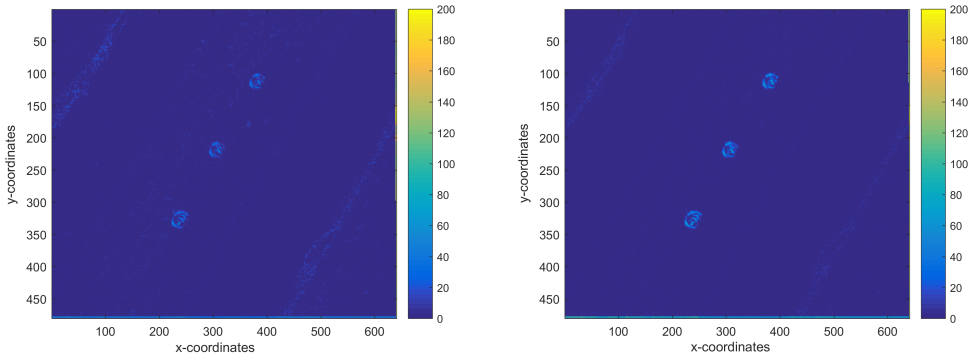
The contribution mask that is introduced in the previous subsection will be used for GME. Since the size of the testing images is 640×480, the contributing area includes the pixels that are in the middle of the image with the size 620×460. Frames 360-361 and Frames 650-651 are still used for testing. The registration errors are shown in Figures 2.9 and 2.10. By comparing these figures with Figures 2.6 and 2.7, the image registration error is obviously reduced. Although the contribution mask is applied, the student-t cost function is still apparently better than the quadratic cost function. Under this condition, the moving objects become the dominate factor

that damages the transformation estimation. The moving objects always introduce high errors when the optimisation nearly reaches the convergence point (they are always highlighted in the figures). As the quadratic cost function magnifies the cost quadratically, it leads the optimisation to the wrong directions with larger steps. However, a robust cost function can reduce the effect. Therefore, the proposed cost function can make an improvement.



(a) Using quadratic cost (Error: 47546) (b) Using student-t cost (Error: 44866)

FIGURE 2.9: Comparing the image registration error between using the quadratic cost and student-t cost on Frames 360-361. The contribution mask is applied.



(a) Using quadratic cost (Error: 60342) (b) Using student-t cost (Error: 47173)

FIGURE 2.10: Comparing the image registration error between using quadratic cost and student-t cost on Frames 650-651. The contribution mask is applied.

The report above suggests using a contribution mask to address the pixels with an invalid source after transformation and using the proposed adaptive robust cost function to address the outliers within the contributing area. This leaves the problem regarding how to choose the size of the contributing area in practice. Usually, we can use a fixed size for an entire video. However, some videos can have large displacements (especially rotation) within a short and unpredictable period of time. Although the robust cost function can resist the problem, we tend to eliminate it using a feasible and simple scenario. Choosing a small contributing area for all the frames is fine, but this would decrease the information for GME that could degrade the accuracy. In practice, we suggest using a gradually decreasing contributing size every time the transformation

error is detected as abnormal. It is a trade-off between time and reliability. In the following experiments, such extreme situations do not appear.

2.3.5.2 General Results

To attain fair results, a contribution mask is used for all cost functions. The contribution mask considers the middle 620×460 area in the 640×480 image. The experiment tests the L1 norm[5], L2 norm, Huber's M-estimator, Tukey's M-estimator, and the proposed student-t cost function on four datasets in the Vivid benchmark. Regarding the parameter, ν, of the student-t cost, we perform the self-adapted algorithm that is described in Section 2.3.2. To create comprehensive results, we decrease the frame rate (by down sampling) to approximately 12 Hz, 6 Hz, and 3 Hz. The problem becomes more complicated with a lower frame rate, as there will be a large displacement, thus, they are easily stuck in local minima.

Table 2.2 shows the average extended L0 norm errors (2.47) from using different cost functions over four datasets. The performances of the cost functions show a significant difference with different datasets and variational frame rates. However, the proposed algorithm provides the most numerous best estimations. Table 2.3 shows a summary of Table 2.2. From the table, the proposed student-t cost with auto-tuning parameters provides nine best and three second best estimations over 12 experiments. It is clearly the best approach among them. The second-best cost function is hard to determine. Tukey's M-estimator yields three best estimations but also three worst estimations. It seems that Huber's M-estimator is more stable so that its 11 estimations are in the first three positions in the ranking. The choice between both depends on whether the input video contains a similar mode of content. The L1 norm and L2 norm are less reliable in the testing videos.

Table 2.4 shows the average iterations to converge using different cost functions. Since only Newton's method is used for minimisation, this metric can reflect the computational cost. As mentioned, the proposed approach aims to perform at a satisfying speed and ensure the best accuracy. It makes sense that the proposed approach does not show superiority on this metric. Hence, we cross-check Table 2.2 with Table 2.4 as follows. For the Vivid 1 dataset, the best two cost functions are Tukey's M-estimator and the student-t cost. The student-t cost outperforms the other at 12 Hz/3 Hz and provides very similar performance at 6 Hz. The computational costs are similar. In terms of Vivid 2, the performance of the L1 norm based approach takes second place, but it is very computationally expensive, while the proposed cost function is much faster and generates the best estimations. Regarding Vivid 3 and 4, both M-estimators yield the second best estimations compared to the student-t cost, but they have no significant advantage in time consumption.

In general, the above experiments show the reliability of the approach, especially when the video content is unpredictable. At the same time, the proposed approach does not increase the converging iterations compared to its competitors in accuracy. The proposed student-t cost

[5]We have not implemented the most sophisticated L1 norm based optimisation algorithms.

Dataset	L1	L2	Huber	Tukey	Student t
EgTest01(12 Hz)	5.48	7.66	6.83	5.13	**5.06**
EgTest01(6 Hz)	9.00	10.01	7.69	**5.49**	5.50
EgTest01(3 Hz)	13.11	12.36	8.15	6.67	**6.29**
EgTest02(12 Hz)	**4.62**	5.18	5.07	6.40	4.69
EgTest02(6 Hz)	5.11	6.32	5.98	6.98	**4.95**
EgTest02(3 Hz)	6.74	7.83	6.43	6.84	**5.37**
EgTest03(12 Hz)	12.28	12.19	12.16	**12.08**	12.10
EgTest03(6 Hz)	13.13	12.75	12.66	12.35	**12.24**
EgTest03(3 Hz)	14.75	14.00	13.68	**12.97**	**12.97**
EgTest4(12 Hz)	9.20	9.03	8.43	10.33	**8.31**
EgTest4(6 Hz)	14.72	11.34	10.32	11.29	**10.00**
EgTest4(3 Hz)	21.10	14.14	12.30	12.51	**11.86**

TABLE 2.2: The mean extended $L0-norm$ errors ($[\times 10^4]$) of image registration over all pairs of images (smaller the better). Sampling rate shown in brackets.

Position in Ranking	Cost Functions (number of experiments on the position)
1	Student t (9), Tukey (3), L1 (1)
2	Huber (4), Student t (3), Tukey (3)
3	Huber (7), Tukey (2), L1 (2), L2 (1)
4	L2 (8), L1 (2), Huber (1), Tukey (1)
5	L1 (6), Tukey (3), L2 (3)

TABLE 2.3: The number of experiments for each cost function in each position in the ranking of Table 2.2. Note that a tie for first place exists on the dataset Vivid 3 (3 HZ).

Dataset	L1	L2	Huber	Tukey	Student t
EgTest01(12 Hz)	30.4	8.2	6.2	15.6	17.3
EgTest01(6 Hz)	22.9	10.8	9.2	16.4	15.9
EgTest01(3 Hz)	23.4	18.3	13.6	22.1	21.6
EgTest02(12 Hz)	37.8	7.8	5.3	32.6	9.6
EgTest02(6 Hz)	31.4	8.6	5.9	42.1	23.6
EgTest02(3 Hz)	33.0	10.7	8.3	43.2	16.1
EgTest03(12 Hz)	21.5	7.7	5.0	14.3	7.1
EgTest03(6 Hz)	17.5	9.2	6.4	19.9	10.4
EgTest03(3 Hz)	17.4	13.0	10.3	23.6	29.5
EgTest04(12 Hz)	23.1	8.2	6.3	26.9	9.1
EgTest04(6 Hz)	25.4	11.9	8.6	27.2	13.5
EgTest04(3 Hz)	23.6	21.4	16.2	32.4	21.1

TABLE 2.4: The mean iteration numbers to converge minimum error.

function is a good substitute for the existing cost function in GME. It can ensure accuracy and maintain relatively high computational efficiency.

2.4 Mosaic Background Subtraction

Mosaic background subtraction combines the proposed GME algorithm and existing background subtraction algorithms to detect moving objects from videos. In the end of this section, an existing multi-target tracker will be applied to the detections. This section presents the feasibility of a complete detect and track system while the scene of the video is flat.

2.4.1 Mosaic Background

After performing the robust GME algorithm for each pair of consecutive frames in a video, we can obtain a chain of transformations that projects any frame to the scene space of the first frame.[6] For modelling the mosaic background, we first choose a codebook-based background subtraction model, ViBe [20] (while [26] is another good option) and use the empirical parameters as follows:

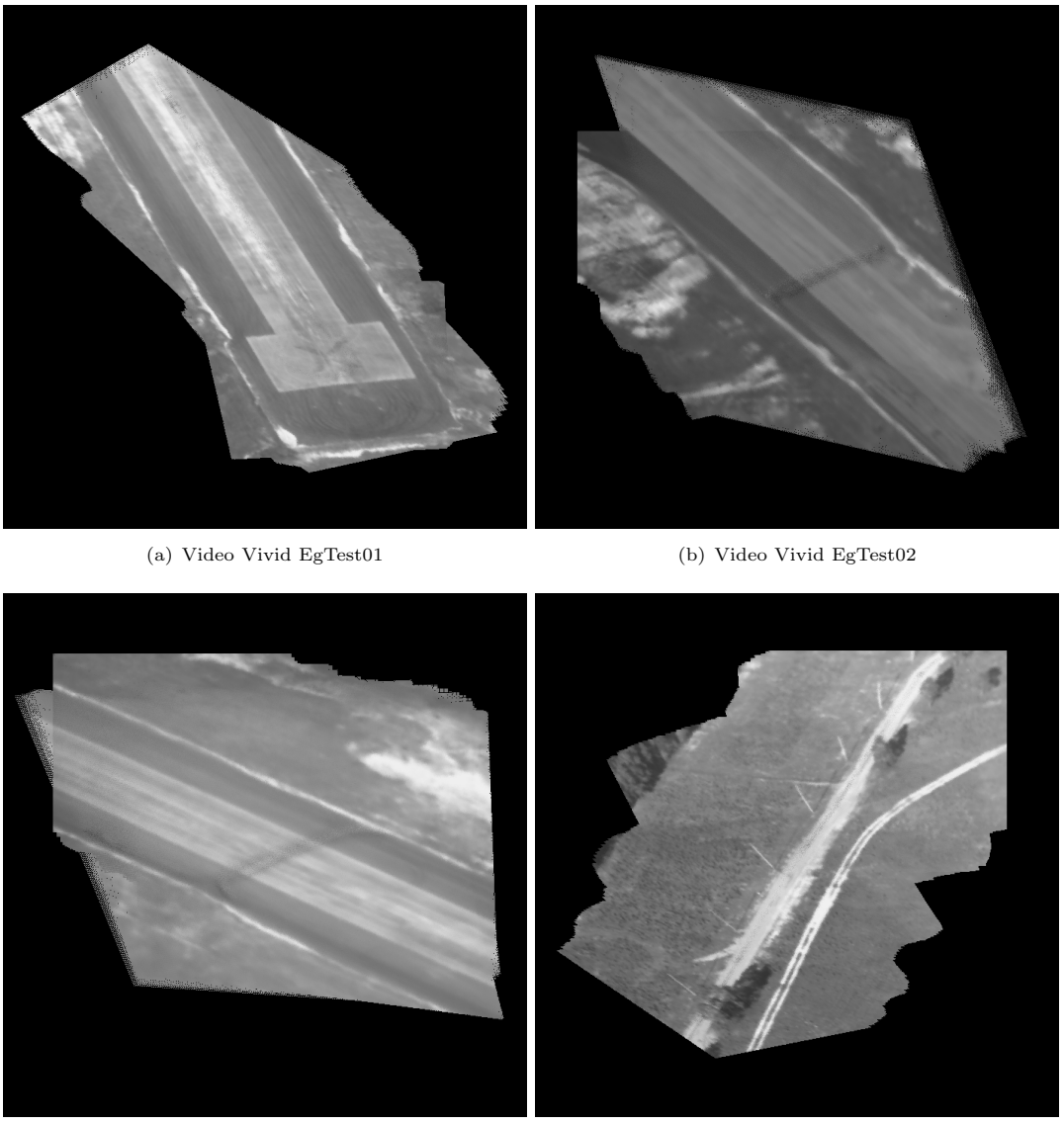
- number of samples per pixel: $N = 15$
- radius of sphere (the threshold for classifying the background): $R = 6$
- number of close samples for being part of the background: $\#_{min} = 2$
- number of random subsampling (for substituting background samples): $\phi = 3$

We then initialise a larger (relative to the fixed resolution of the video) empty background model of ViBe. The first frame is used for initialising the corresponding area[7] in the canvas. The following frames will be projected to this canvas via the chain of transformation matrices. If the pixels in the transformed frames correspond to an empty area in the canvas, these pixels will be used to initialise this area and will not be considered background for several iterations. If the pixels are projected into initialised area, those pixels will be updated according to ViBe.

Figure 2.11 shows the mosaic background models of the datasets, EgTest01-04. Since there is no ground-truth for the mosaic background, the background models can only be evaluated by inspecting the raw video. The background for EgTest01 reflects the true mosaic scene and includes major visual features in the video. The background for EgTest02 and 03 are less accurate. The backgrounds are blurred, and the road areas are not naturally stitched (see Figures 2.11(b) and 2.11(c)). The reason is that both datasets contain less textures for GME. The scene is not flat such that the affine transformation matrices are approximated, and the errors accumulate amid the chain of transformations. EgTest04 contains bumpy scenes and several continuous out-of-focus frames. These issues lead to inaccurate transformation matrices within the chain, thus influencing the entire mosaic. Moreover, the appearance of street lights is obviously faulty, which is caused by the scene depth and different viewpoints.

[6] We can project any frame to the scene of any frame with the transformations, but in order to provide a reader-friendly visualisation, we project all the frames to the scene of the first frame.

[7] We put the initial frame on a suitable place of the canvas, considering the resolution and the overall transformations, so any area in the following frames will not be projected to somewhere outside of the canvas.



(a) Video Vivid EgTest01

(b) Video Vivid EgTest02

(c) Video Vivid EgTest03

(d) Video Vivid EgTest04

FIGURE 2.11: Estimated mosaic background.

2.4.2 Moving Object Detection

The moving object detection is therefore straightforward. Figures 2.12-2.15 show several examples of detection results on the EgTest01-04 datasets. In Figure 2.12, most of the moving vehicles are successfully detected. We can see a few missing or incomplete detections in Frames 540 and 620. The reason is that the top areas of the frames are always newly involved due to the camera motion, such that they are not considered background. There is a large false alarm in Frame 460. By inspecting the video, this area appears in the early frames but is not in the view for a long period. When it is revisited, the light condition changes; thus, it is classified as foreground.

In Figure 2.13, mis-detection is rare, and there are some false alarms off the road. That

is the area in which the scene is not flat. When the vehicles are close to each other, extended detections occur, though tracking them is not the focus of this thesis.

Figure 2.14 is similar to Figure 2.13, except that the colours of the vehicles are more similar to the background. Nevertheless, our algorithm detects most of them. The cause of the large false alarms in Frames 620 and 700 is the same as those in Figure 2.12 in which revisiting occurs a long period afterwards.

The background is more complicated in Figure 2.15, and the altitude of the airborne camera is obviously reduced. Frame 220 is one of the out-of-focus images, but it is not influenced significantly. Figure 2.16 shows a detailed example of the detection results when the focus loses and recovers. Note that we process every five frames during the GME rather than frame by frame. This figure shows only after a couple of iterations. The algorithm can generate correct results after extreme blurred images. This situation reflects one advantage of the pixel-based GME compared to feature-based approaches.

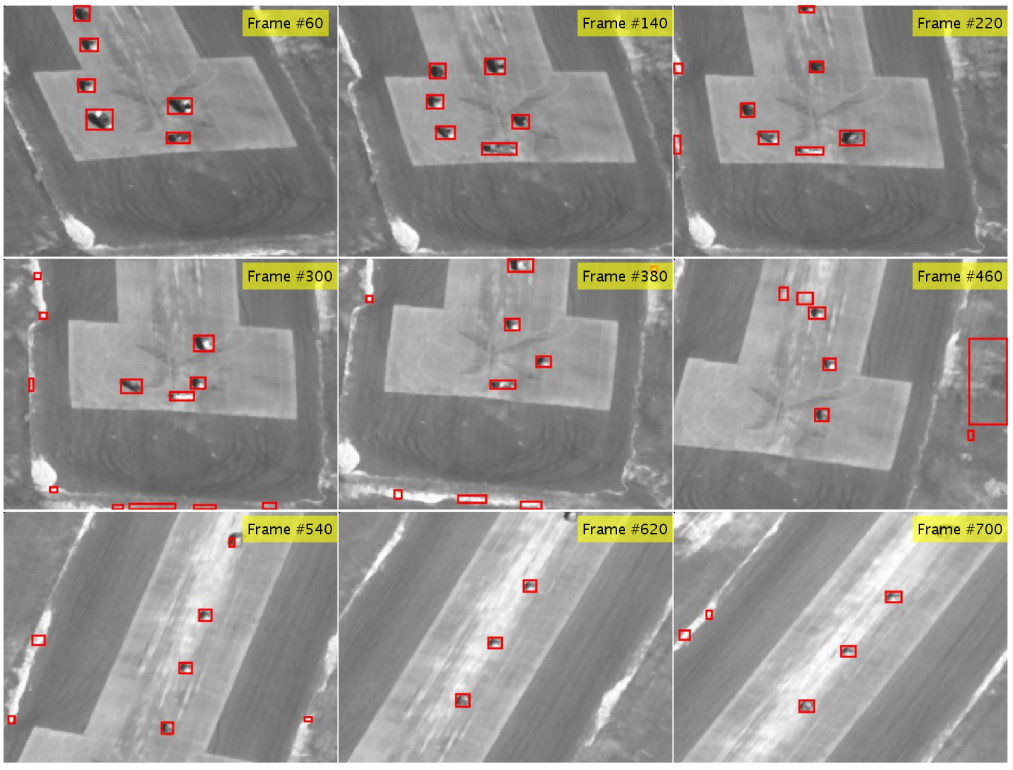


FIGURE 2.12: Detection results on Vivid EgTest01.

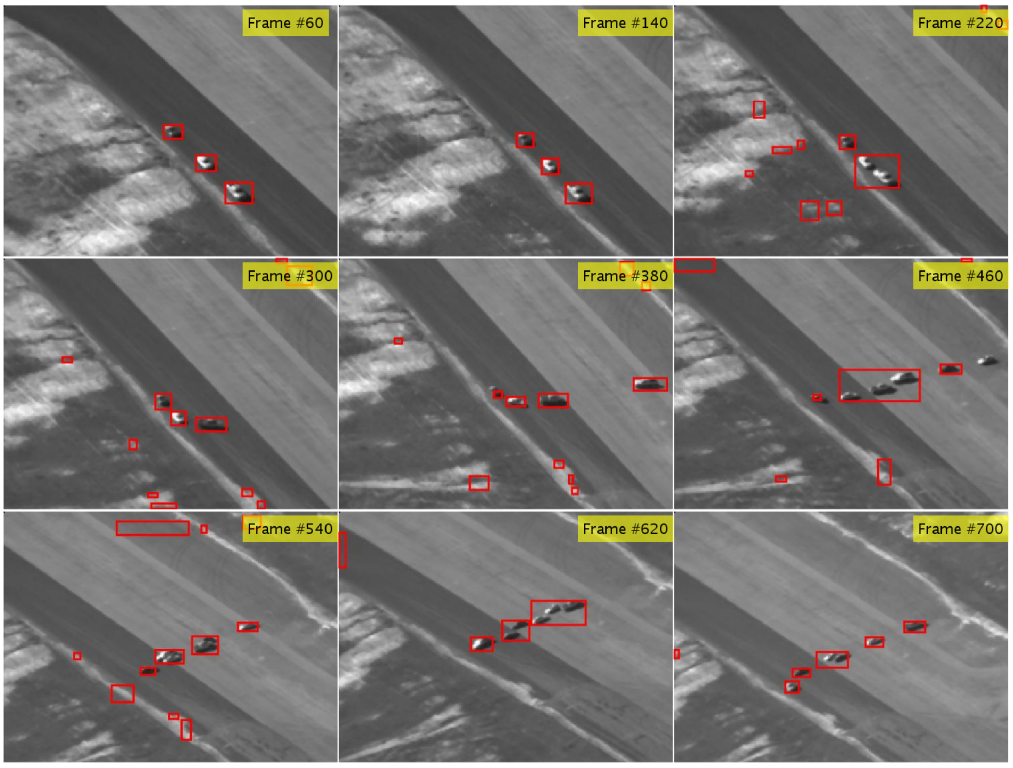


FIGURE 2.13: Detection results on Vivid EgTest02.

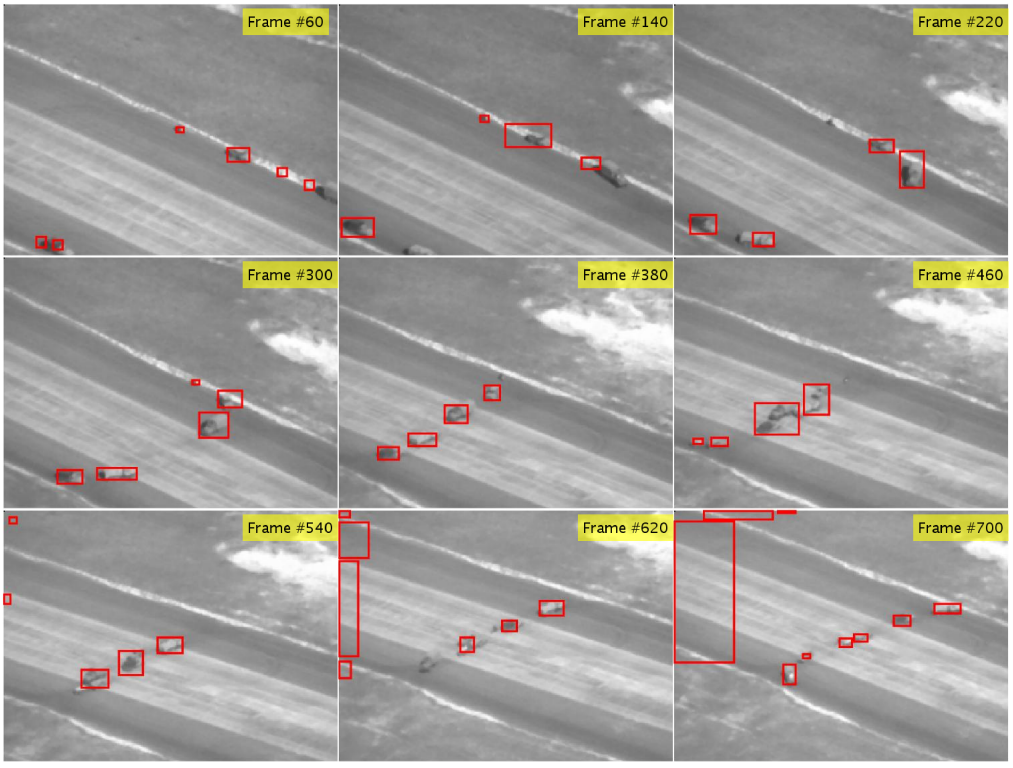


FIGURE 2.14: Detection results on Vivid EgTest03.

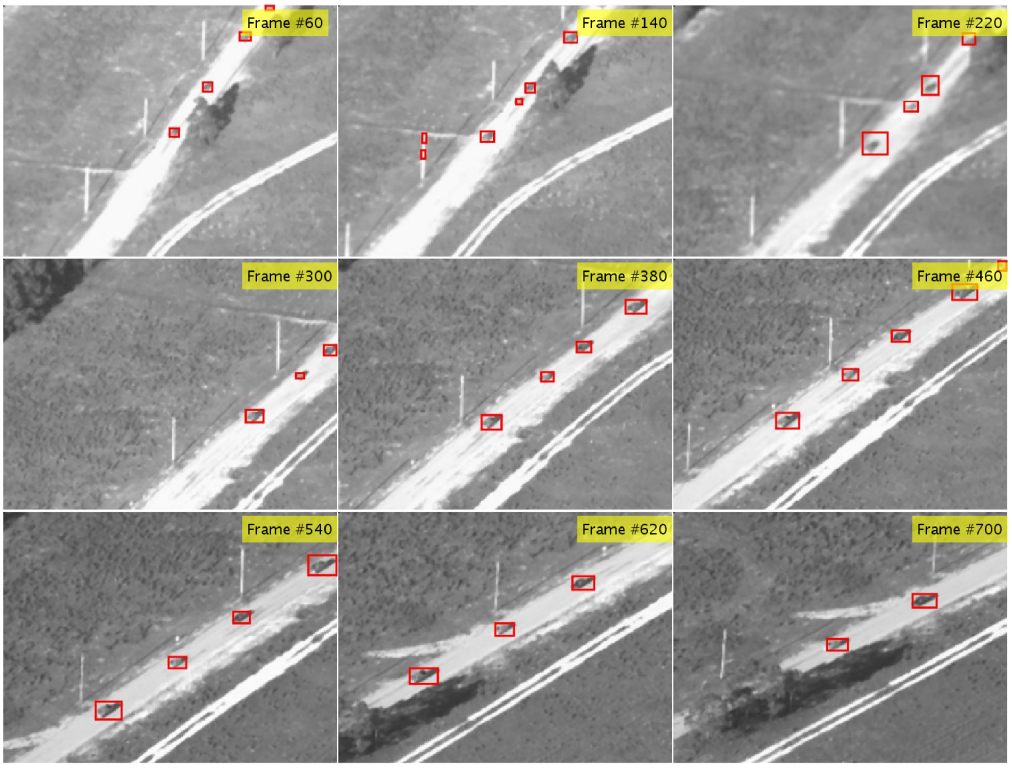


FIGURE 2.15: Detection results on Vivid EgTest04.

FIGURE 2.16: The moving object detection recovers from out-of-focus frames.

2.4.3 Multi-target Tracking

A Gaussian Mixture Probabilistic Hypothesis Density (GM-PHD [77]) filter has been used to analyse the detections on Vivid EgTest01 and Vivid EgTest02. The essential parameters for the GM-PHD filter are displayed in Table 2.5.

Parameter	Value	Information
d_t	1	The time interval (It is not significant to use frame rate).
ν_{init}	10	If a measurement's velocity is smaller than this, a track will be initialise based on the measurement.
μ_{init}	$[0;0;0;0]^T$	Initial state for each new target.
Σ_{init}	$diag([10;10;5;5]^2)$	Initial covariance of the state for each new target.
η_{merge}	3	If the Mahalanobis distance between two tracks is smaller than this value, they will be merged.
$\eta_{display}$	0.8	If the weight of the track is greater than this value, it will be displayed.
η_{del}	0.05	If the weight of the track is smaller than this value, it will be deleted.
P_d	0.75	Probability of detection. It is an empirical value in this case.
Q	Equation (3.6) when $D = 2$ and $\sigma = 1$	Process noise.
R	$diag([5,5]^2)$	Measurement noise.
$n_{clutter}$	$poissrnd(5)$	The number of clutters. It is not a significant value in this case.
Target spawning is not considered in these detections.		

TABLE 2.5: The essential parameters for GM-PHD filter while tracking the detections that are produced in Section 2.4.2.

The results are shown in Figures 2.17 and 2.18. For the relatively simpler dataset (EgTest01), targets #1 and #20 are correctly preserved during almost the entire video, and other targets are tracked well for at least hundreds of frames. In terms of EgTest02, it is more complicated due to the extended targets. The tracking results are satisfying, considering that the multi-target tracker is not meticulously optimised and that the imagery features are not used during the filtering process.

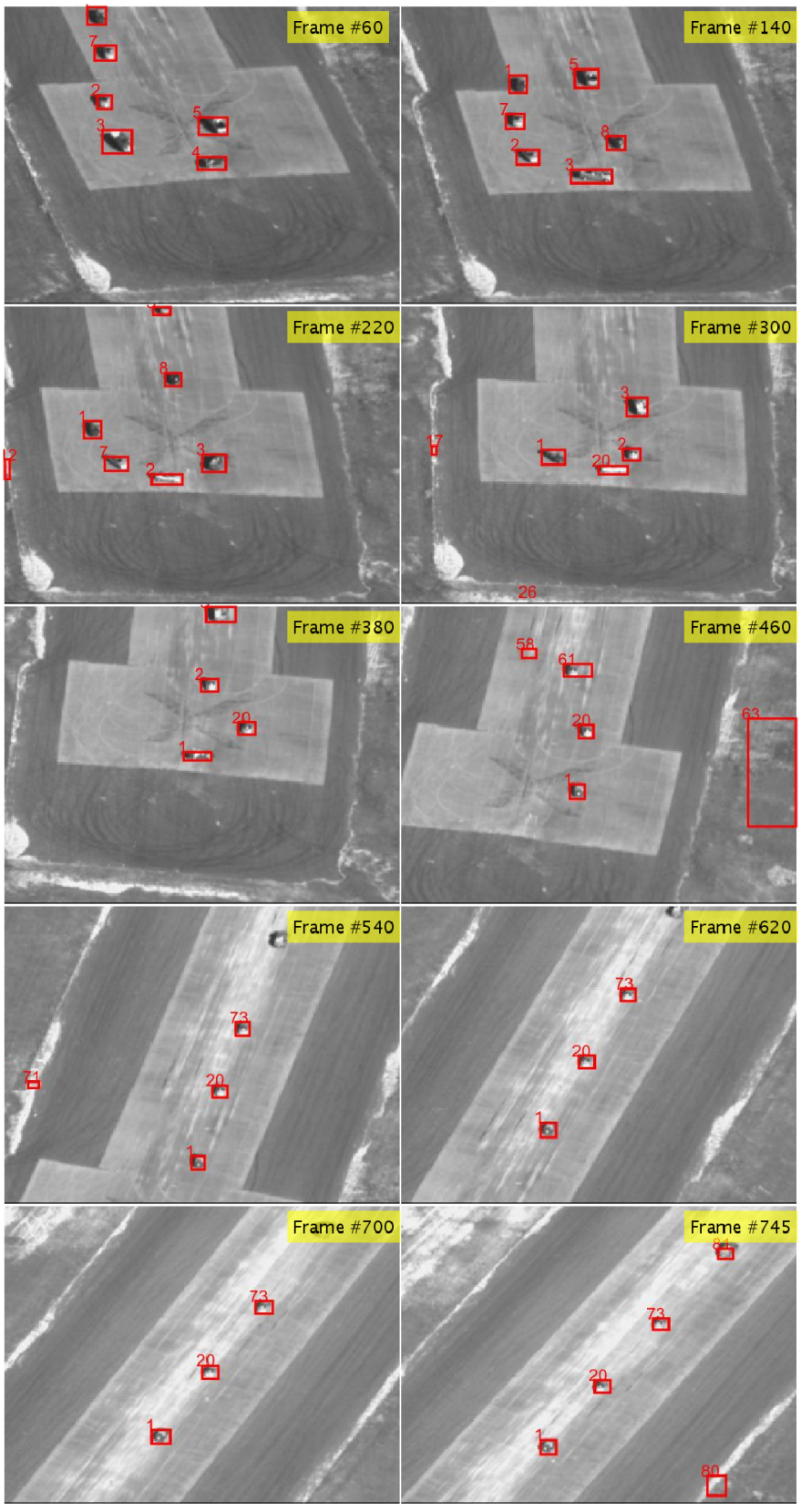


FIGURE 2.17: Tracking results on Vivid EgTest01.

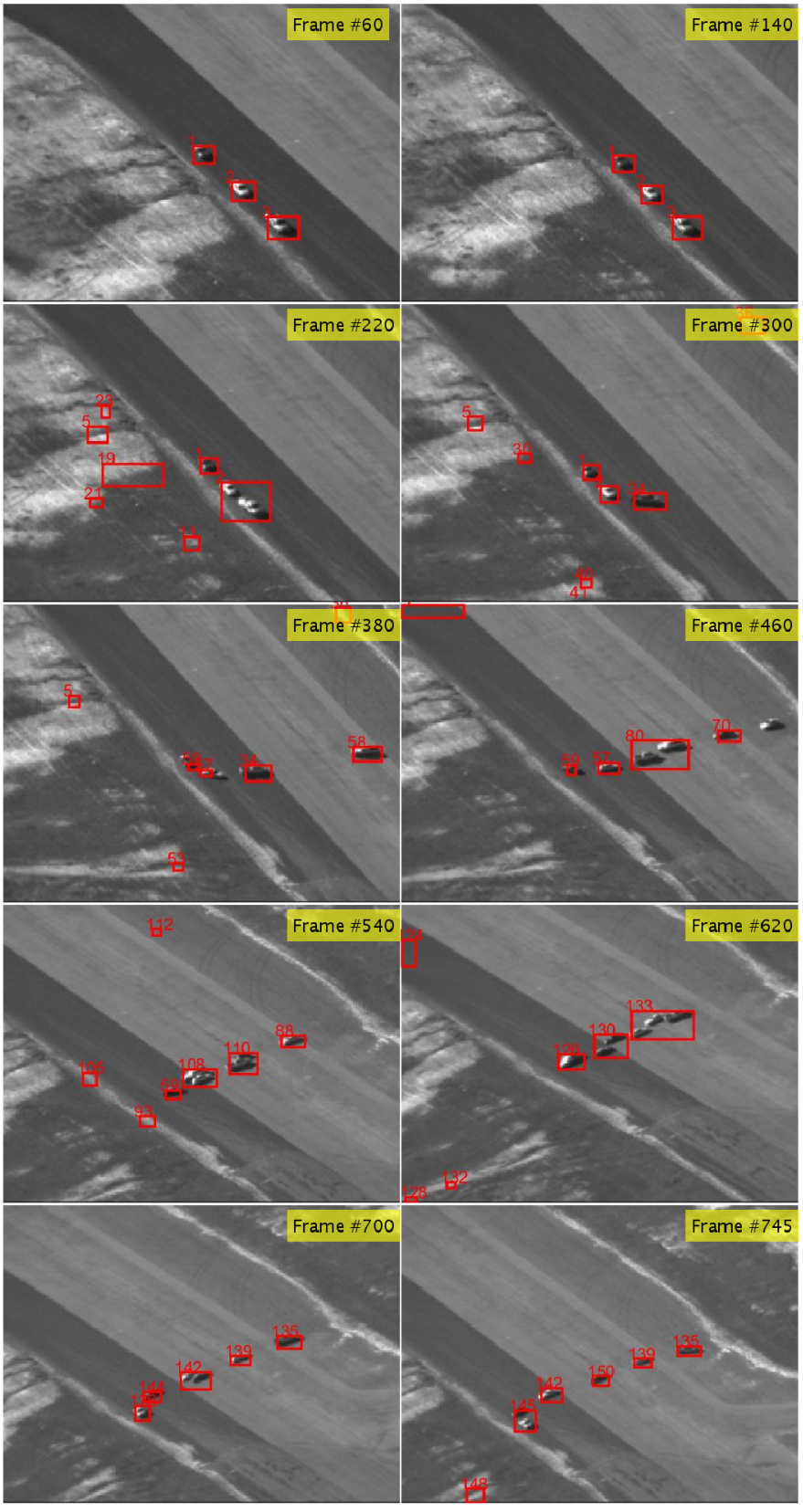


FIGURE 2.18: Tracking results on Vivid EgTest02.

2.5 Conclusions

This chapter has described several moving object detection algorithms for a moving camera with an assumption that the scene is flat. A new adjustable and robust cost function, which is derived from student t-distribution, is proposed for pixel-based GME. This student-t cost function can perform as the interpolation between Cauchy function and L2 norm given different parameters. We also proposed a re-parameterisation method that the parameters are more explicit for GME purpose on videos. Comparisons have been made with existing cost functions on a public benchmark. The results indicate that proposed cost function can trade-off between robustness and efficiency during a video with a proposed parameter estimation algorithm. In order to show the feasibility of generating a complete processing chain for processing videos, the proposed GME approach has been used to create mosaic backgrounds with ViBe [20]. Background subtraction is used for detecting moving objects in the videos. Afterwards, the detection results are reported and feed to a multi-target tracker, GM-PHD filter [77]. The tracking results are finally presented.

Chapter 3

Particle Filtering Using Near-Optimal Proposal and Rao-Blackwellisation

3.1 Introduction

The Kalman filter will always be the most effective and accurate method for processing an incoming data stream for systems that are truly linear and Gaussian [78]. However, real problems rarely involve systems that are truly linear and Gaussian. Particle filters can exploit non-linear and non-Gaussian state-space models to tackle such real problems. This contrasts with algorithms such as the extended Kalman filter (EKF) and unscented Kalman filter (UKF) [79], which involve approximation with models that are locally linear and Gaussian. When the information that can be captured by non-linear and non-Gaussian models is significant relative to the information that can be captured by linear and Gaussian models, a particle filter will often outperform the EKF and UKF. Scenarios in which this is the case occur in a variety of applications and, as a result, particle filters have been applied successfully in numerous contexts.

Given this widespread interest, it is unsurprising that many improvements to particle filters have been proposed. The two of specific interest in this chapter are the use of a near-optimal proposal and the use of Rao-Blackwellisation.

As will be explained in more detail later in the chapter, a particle filter manipulates randomly-generated particles (i.e., hypotheses for the state of the system). The proposal distribution is used to generate these particles. It matters that the particles are generated randomly. However, while it is possible (and common) to generate samples that are consistent with the anticipated time evolution of the system without any reference to the observed data, it can be inefficient to do so. It can therefore be advantageous (in terms of the estimation accuracy that can be achieved with a given amount of computation) if the random process generating the samples attempts to put particles in sensible places (e.g., that are consistent with the observed data). This concept is formalised by the definition of an optimal proposal distribution [80, 81]. In many scenarios, while it is possible to define the optimal proposal mathematically, it cannot be sampled; thus, near-optimal proposals need to be used. Such near-optimal proposals can use, for example, similar approximations to those used in an EKF or UKF [82–84]. It is also

possible to define a proposal using state-space transformations [85]. Such transformations have been developed to help the EKF to work well in certain geometries.

Near-optimal proposals are typically more computationally expensive per particle than simply sampling in a way that is blind to the measurement. As a result, near-optimal proposals are useful when the advantage of putting the particles in a sensible place offsets the additional computational expense of doing so. Put simply, there are scenarios in which using a near-optimal proposal is not worth it.

Rao-Blackwellisation is an idea conceived by Rao and Blackwell [86, 87]. The generic idea is exemplified by separating an estimation problem into two constituent sub-problems such that one sub-problem is considered conditional on a sample related to the other sub-problem. This decomposition can be advantageous if knowing the sample related to one sub-problem makes it easier to consider the remainder of the estimation task. This idea can be applied in the context of particle filters. This thesis considers the special case[1] in which a particle filter is used for one sub-problem such that the other sub-problem involves models that are (conditionally) linear and Gaussian. Since the second sub-problem is then linear and Gaussian, a Kalman filter can be used (for each particle). Such Rao-Blackwellisation has been considered in a particle filter context in the past [21, 81, 88–93]. Other other side, a case should be highlighted which considers the correlation of the process noises that is considered by both the Kalman filter and particle filter. A generic description has been described in [21, 93].

In an analogous way to the near-optimal proposals, using Rao-Blackwellisation typically increases the computational load per particle. Therefore, there is a need to understand when Rao-Blackwellisation is worth using.

Both near-optimal proposals and Rao-Blackwellisation have been the subject of research for many years, and descriptions of the combination of the two exist (e.g., [21, 94]). As an application, the combination is considered widely in the context of simultaneous localisation and mapping (SLAM). The motivation of using both in SLAM is two-fold: the dimensionality of the state-space that would need to be considered if a particle filter without Rao-Blackwellisation were applied and the need to use a near-optimal proposal to ensure that the observed data influences the hypotheses considered for robot motion. This combination was first described in [22] as FastSLAM 2.0 (prior to FastSLAM 2.0, FastSLAM [95] used Rao-Blackwellisation but did not use a near-optimal proposal). It is perceived that the existing literature in different disciplines (e.g., [96–99]) has seldom tried to use the combination of Rao-Blackwellisation and near-optimal proposals in the presence of correlated noise. This situation has been described in [21]. However, we are unaware of papers that attempt to ascertain when, empirically, the combination of techniques has merit over using each in isolation.

This chapter's contribution is two-fold. A generically applicable description of a particle filter will be provided that is carefully described such that it can use both a near-optimal proposal and Rao-Blackwellisation with correlated process noise. The performance of particle filters will be analysed using all subsets of the near-optimal proposal and Rao-Blackwellisation with the

[1] Other special cases exist, e.g., involving a hidden Markov model operating in conjunction with a particle filter.

aim of understanding when the techniques (alone and in combination) provide advantages in terms of the trade-off between computational cost and estimation accuracy.

The organisation of this chapter is as follows. Section 3.2 describes the problems that this chapter will discuss. Section 3.3 briefly introduces the theory of importance sampling. Section 3.4 introduces the generic particle filter. It then includes two existing significant techniques (i.e., optimal proposal and Rao-Blackwellisation) for boosting the performance of a particle filter. The particle filter combining such two techniques is describe in Section 3.4.4 and evaluated in Section 3.4.5. Finally, 3.5 presents the conclusions of this chapter.

3.2 Problem Statements

This section includes generic definitions of linear and non-linear Gaussian problems. Near-constant velocity model and two-dimensional range-bearing model are described as examples. Both of them will be used in the experiments.

3.2.1 Linear Gaussian Problems

3.2.1.1 Definition

A model of the linear Gaussian problem is defined as follows.

The dynamic model is described as:

$$x_t = F \cdot x_{t-1} + \omega_t, \tag{3.1}$$

$$\omega_t \sim \mathcal{N}(0, Q), \tag{3.2}$$

where x_t and x_{t-1} are the current and previous states, F is the state transition matrix, and ω_t is the process noise defined by a zero-mean Gaussian white noise with covariance Q. It is known to the author that a more general dynamic model should consider the control parameters. However, because the cases that are discussed in this chapter do not involve the control input, we only consider the cases in which the control parameters are either included in the state vector or are explicit to the system. Such shorter dynamic models will also be used in the rest of this chapter.

The measurement model is defined as follows:

$$z_t = H \cdot x_t + \epsilon_t, \tag{3.3}$$

$$\epsilon_t \sim \mathcal{N}(0, R), \tag{3.4}$$

where z_t is the observation of the system, H is the measurement model, and ϵ_t is the measurement noise which is defined by a zero-mean Gaussian white noise with covariance R.

3.2.1.2 Near-constant Velocity Model

A general nearly constant velocity model[2] [100] can be described as follows:

$$x_t = F \cdot x_{t-1} + Q_t{}^{\frac{1}{2}} \omega_t, \tag{3.5}$$

where

$$\omega_t \sim N\left(\mathbb{O}_{2D\times 1}, \mathbb{I}_{2D\times 2D}\right), \tag{3.6}$$

where $\mathbb{I}_{D\times D}$ is an identity matrix with dimension $D \times D$ and $\mathbb{O}_{D\times D}$ is a zero matrix with dimension $D \times D$. We assume that $x_t = [x_{t,1}, x_{t,2}]^T$ such that $x_{t,1}$ and $x_{t,2}$ are vectors representing the position and velocity, respectively. The dimension of $x_{t,1}$ and $x_{t,2}$ depends on the problem (e.g., whether the state is 2D or 3D). We assume we can decompose F as follows:

$$F = \begin{bmatrix} \mathbb{I}_{D\times D} & \Delta T \cdot \mathbb{I}_{D\times D} \\ \mathbb{O}_{D\times D} & \mathbb{I}_{D\times D} \end{bmatrix}, \tag{3.7}$$

where ΔT is the (continuous) time between epochs $t-1$ and t. In addition, Q_t parameterises the uncertainty of the position and the correlated uncertainty in the velocity:

$$Q_t = \sigma \begin{bmatrix} \frac{1}{3}\Delta T^3 \cdot \mathbb{I}_{D\times D} & \frac{1}{2}\Delta T^2 \cdot \mathbb{I}_{D\times D} \\ \\ \frac{1}{2}\Delta T^2 \cdot \mathbb{I}_{D\times D} & \Delta T \cdot \mathbb{I}_{D\times D} \end{bmatrix}, \tag{3.8}$$

where σ quantifies the amount of (white noise) acceleration assumed. Please see [100] for more details.

3.2.2 Non-linear Problems

3.2.2.1 Definition

We assume a non-linear problem as follows.

The dynamic model is as follows:

$$x_t = f(x_{t-1}) + \omega_t, \tag{3.9}$$

$$\omega_t \sim \mathcal{N}\left(0, Q\right), \tag{3.10}$$

[2] We use x to denote the state vector since it is general and can be for Cartesian, angular, etc., spaces.

where x_t and x_{t-1} are the current and previous states, $f(.)$ is a non-linear state transition function, and ω_t is the process noise defined by a zero-mean Gaussian white noise with covariance Q_t.

The measurement model is as follows:

$$z_t = h(x_t) + \epsilon_t, \tag{3.11}$$

$$\epsilon_t \sim \mathcal{N}(0, R), \tag{3.12}$$

where z_t is the observation of the system, $h(.)$ is a non-linear measurement model, and ϵ_t is the measurement noise defined by a zero-mean Gaussian white noise with covariance R_t. Note that a non-linear problem can be one of three cases: a non-linear dynamic with a linear measurement model, a linear dynamic with a non-linear measurement model, and a non-linear dynamic with a non-linear measurement model.

3.2.2.2 Range-Bearing Measurement Model

A typical non-linear model is the range-bearing model widely used by radar, lidar, and sonar sensors. The relation between a radar measurement (include range and bearing observations) and the target location in 2D Cartesian world is calculated as follows:

$$h(x_t) = \begin{bmatrix} \sqrt{(X_t - \breve{X})^2 + (Y_t - \breve{Y})^2} \\ arctan((Y_t - \breve{Y}), (X_t - \breve{X})) \end{bmatrix} + \epsilon_t, \tag{3.13}$$

where $\left[\breve{X}, \breve{Y}\right]^T$ is the coordinate (on the x-axis and y-axis) of the sensor in the Cartesian world, $[X_t, Y_t]^T$ is the coordinate of the target at time t, and the measurement noise covariance is as follows:

$$R = \begin{bmatrix} \sigma_R^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}. \tag{3.14}$$

3.3 Importance Sampling

Particle filters are an application of importance sampling. Therefore, before explaining the particle filter in detail, we describe importance sampling (more details can be found in [101]).

3.3.1 Description

Suppose the task is to make inferences about a state, x, given a *target* distribution,[3] $\pi(x)$. The core idea of importance sampling is to make such inferences given samples from a different distribution, a *proposal* distribution, $q(x)$. As will be shown, this has the advantage that it only requires that $\pi(x)$ can be evaluated point-wise up to normalisation (i.e., it is not required that $\int \pi(x)\, dx = 1$). We assume that the inference of interest can be expressed as follows:[4]

$$\bar{f} = \int f(x)\, \pi(x)\, dx, \tag{3.15}$$

such that

$$\bar{f} = \int f(x)\, \pi(x)\, \frac{q(x)}{q(x)} dx \approx \hat{f}, \tag{3.16}$$

since $\frac{q(x)}{q(x)} = 1$ and where

$$\hat{f} = \frac{1}{N} \sum_{i=1}^{N} f\left(x^i\right) w^i \qquad x^i \sim q(x), \tag{3.17}$$

where

$$w^i = \frac{\pi\left(x^i\right)}{q\left(x^i\right)}, \tag{3.18}$$

such that we approximate the integral of interest, $\bar{f}$, as a weighted sum over a set of N samples drawn from $q(x)$.

To understand some of the considerations that are important when designing proposal distributions, it is helpful to consider the expectation and variance of the importance sampling approximation, $\hat{f}$, as a function of $q\left(x^{1:N}\right)$, the joint distribution of the N samples that are drawn from the proposal distribution. The expectation can be shown to be equal to the integral

[3] Although if x is continuous, we should technically refer to densities not distributions, we have tried to avoid being perceived as pedantic and simply referred to 'distributions' throughout.

[4] In a particle filter and in other contexts, this argument is used with $\pi(x) = p(x, y)$ for a fixed y such that $\pi(x)$ does not integrate to unity. In that setting, (3.15) is not an expectation of f with respect to $p(x|y)$. However, we can approximate that expectation by noting that it is equal to $\frac{\int f(x)\pi(x)dx}{\int \pi(x)dx}$ and then by approximating both the numerator and denominator using the techniques that are described in (3.15)-(3.18). The statistical accuracy of the resulting ratio of estimators is more complex to analyse than that considered in the appendix and is not discussed further here. The purpose of this discussion is primarily to provide an intuitive explanation of the rationale for the discussion that follows (3.33).

of interest:

$$\mathbb{E}_{q(x^{1:N})}\left[\hat{f}\right] = \mathbb{E}_{q(x^{1:N})}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{\pi(x^i)}{q(x^i)}f(x^i)\right] \tag{3.19}$$

$$= \int\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\pi(x^i)}{q(x^i)}f(x^i)\right)q(x^{1:N})dx^{1:N} \tag{3.20}$$

$$= \int\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\pi(x^i)}{q(x^i)}f(x^i)\right)\prod_{i=1}^{N}q(x^i)dx^{1:N} \tag{3.21}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\Big(\underbrace{\int\frac{\pi(x^i)}{q(x^i)}f(x^i)q(x^i)dx^i}_{\bar{f}} \times \underbrace{\int\frac{\prod_{j=1}^{N}q(x^j)}{q(x^i)}dx^1...dx^N}_{Unity}\Big) \tag{3.22}$$

$$= \bar{f}, \tag{3.23}$$

with a variance that can be calculated to be as follows:

$$\sigma^2 = \mathbb{E}_{q(x^{1:N})}\left[(\bar{f}-\hat{f})^2\right] \tag{3.24}$$

$$= \mathbb{E}_{q(x^{1:N})}\left[\bar{f}^2 - 2\bar{f}\hat{f} + \hat{f}^2\right] \tag{3.25}$$

$$= \bar{f}^2 - 2\mathbb{E}_{q(x^{1:N})}\left[\hat{f}\right]\bar{f} + \mathbb{E}_{q(x^{1:N})}\left[\hat{f}^2\right] \tag{3.26}$$

$$= \mathbb{E}_{q(x^{1:N})}\left[\hat{f}^2\right] - \bar{f}^2 \tag{3.27}$$

$$= \int\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\pi(x^i)}{q(x^i)}f(x^i)\right)^2\prod_{i=1}^{N}q(x^i)dx^{1:N} - \bar{f}^2 \tag{3.28}$$

$$= \int\frac{1}{N^2}\left(\sum_{i=1,i\neq j}^{N}\sum_{j=1}^{N}\frac{\pi(x^i)}{q(x^i)}f(x^i)\frac{\pi(x^j)}{q(x^j)}f(x^j) + \sum_{i=1}^{N}\frac{\pi(x^i)^2}{q(x^i)^2}f(x^i)^2\right)\cdot\prod_{i=1}^{N}q(x^i)dx^{1:N} - \bar{f}^2 \tag{3.29}$$

$$= \frac{1}{N^2}\left(\sum_{i=1,i\neq j}^{N}\sum_{j=1}^{N}\underbrace{\int\frac{\pi(x^i)}{q(x^i)}f(x^i)q(x^i)dx^i}_{\bar{f}}\cdot\underbrace{\int\frac{\pi(x^j)}{q(x^j)}f(x^j)q(x^j)dx^j}_{\bar{f}}\right)$$
$$+\sum_{i=1}^{N}\int\frac{\pi(x^i)^2}{q(x^i)^2}f(x^i)^2q(x^i)dx^i) - \bar{f}^2 \tag{3.30}$$

$$= \frac{1}{N^2}\left((N^2-N)\bar{f}^2 + \sum_{i=1}^{N}\int\frac{\pi(x^i)^2}{q(x^i)}f(x^i)^2dx^i\right) - \bar{f}^2 \tag{3.31}$$

$$< \frac{1}{N^2}\sum_{i=1}^{N}\underbrace{\int c\pi(x^i)f(x^i)^2dx^i}_{c\mathbb{E}[f(x)^2]} \tag{3.32}$$

$$= \frac{c}{N}\mathbb{E}\left[f(x)^2\right], \tag{3.33}$$

where $\frac{\pi(x)}{q(x)} < c$.

It is noteworthy that (3.33) highlights that the variance of an estimate based on importance sampling is scaled as $\frac{1}{N}$, whatever the dimensionality of x. In fact, x does not even have to have a fixed dimension; $\pi(x)$ can describe uncertainty over the dimensionality of x and its value. However, this does not indicate that the performance of importance sampling is independent of dimension. When the dimensionality of the space increases, the mismatch between $\pi(x)$ and $q(x)$ typically increases, forcing c to increase.

3.3.2 The Choice of Proposal Distribution

For the variance to be finite, c must be finite. For c to be finite, there cannot be a value of x for which $\frac{\pi(x)}{q(x)} > c$. Considering what happens as $|x| \to \infty$, it is clear that, for the variance to be finite, the proposal distribution must be more heavy-tailed than the target. It cannot be the case that $\pi(x)$ approaches zero more slowly than $q(x)$. Empirically, when this constraint is violated, no matter how many samples are considered, there are always a very few samples with large weights, and these dominate the estimate in (3.17).

Given that the variance is finite, we wish to minimise this variance. By examining (3.32), it is evident that doing so is equivalent to minimising the following:

$$\sigma_i^2 = \int \frac{\pi\left(x^i\right)^2 f\left(x^i\right)}{q\left(x^i\right)} dx^I, \tag{3.34}$$

subject to $\int q\left(x^i\right) dx^i = 1$ such that we can introduce a Lagrangian as follows:

$$\tilde{\sigma}_i^2 = \int \frac{\pi\left(x^i\right)^2 f\left(x^i\right)}{q\left(x^i\right)} dx^i - \lambda \int q\left(x^i\right) dx^i. \tag{3.35}$$

Setting $\frac{d\left[\tilde{\sigma}_i^2\right]}{dq(x^i)} = 0$ soon leads to:

$$q_{opt}(x) \propto \pi(x), \tag{3.36}$$

where $q_{opt}(x)$ is the optimal proposal.

3.4 Particle Filter

Particle filters build on importance sampling but consider a specific family of target distributions [81]. This family of targets relates to a state, x_t, that evolves with time, t, and is indirectly

observed via a measurement, z_t. The related state-space model[5] is then as follows:

$$x_t = Fx_{t-1} + \omega_t \tag{3.37}$$

$$z_t = h(x_t) + \epsilon_t, \tag{3.38}$$

where ω_t and ϵ_t are the process noise and measurement noise, respectively, where

$$p(\omega_t) = \mathcal{N}(0, Q) \tag{3.39}$$

$$p(\epsilon_t) = \mathcal{N}(0, R), \tag{3.40}$$

where $\mathcal{N}(x; \mu, \Sigma)$ denotes a multivariate normal distribution for x with a mean of μ and a covariance matrix of Σ.

Particle filters use the concept of sequential importance sampling to numerically approximate estimates of interest using a set of (weighted) particles. The target probability is then defined on the joint state, $x_{1:t}$, as

$$\pi(x_{1:t}) = p(x_{1:t}, z_{1:t}) \tag{3.41}$$

$$= p(x_{1:t})p(z_{1:t}|x_{1:t}) \tag{3.42}$$

$$= p(x_1) \prod_{t'=2}^{t} p(x_{t'}|x_{t'-1}) \prod_{t'=1}^{t} p(z_{t'}|x_{t'}) \tag{3.43}$$

$$= p(x_{1:t-1}, z_{1:t-1})\, p(x_t|x_{t-1})p(z_t|x_t). \tag{3.44}$$

The proposal is similarly defined as:

$$q(x_{1:t}|z_{1:t}) = q(x_1|z_1) \prod_{t'=2}^{t} q(x_{t'}|x_{t'-1}, z_{t'}) \tag{3.45}$$

$$= q(x_{1:t-1}|z_{1:t-1})\, q(x_t|x_{t-1}, z_t), \tag{3.46}$$

where $q(x_t|x_{t-1}, z_t)$ is the incremental proposal used to generate the samples of x_t at each time step.

This construction is such that:

$$w_t^i = w_{t-1}^i \frac{p(x_t^i|x_{t-1}^i)p(z_t|x_t^i)}{q(x_t^i|x_{t-1}^i, z_t)}. \tag{3.47}$$

Note that, while the derivation of (3.47) considers the entire history, $x_{1:t}$, an implementation need only store the most recent state, x_{t-1}, for each particle (and the particle weight).

The weights are defined as a mechanism for approximating estimates with respect to a target, $\pi(x_{1:t}) = p(x_{1:t}, z_{1:t})$. To derive estimates with respect to $p(x_{1:t}|z_{1:t}) = \frac{p(x_{1:t}, z_{1:t})}{p(z_{1:t})}$, we

[5] While a particle filter can consider non-linear dynamics, since we are interested in Rao-Blackwellisation in the subsequent sections, we consider linear dynamics throughout this chapter.

note that:

$$p\left(z_{1:t}\right) = \int p\left(x_{1:t}, z_{1:t}\right) dx_{1:t} \approx \frac{1}{N} \sum_{i=1}^{N} w_t^I, \tag{3.48}$$

such that an estimate can be approximated using normalised weights:

$$\int f\left(x_{1:t}\right) p\left(x_{1:t}|z_{1:t}\right) dx_{1:t} \approx \sum_{i=1}^{N} \tilde{w}_t^i f\left(x_{1:t}^i\right), \tag{3.49}$$

where the normalised weights are defined as follows:

$$\tilde{w}_t^i = \frac{w_t^i}{\sum_{i'=1}^{N} w_t^{i'}}. \tag{3.50}$$

While one could iteratively apply sequential importance sampling, over time, the normalised weights will always eventually become skewed; one particle eventually dominates the sum in (3.49). To address this, resampling is introduced. Resampling replaces the current population of particles with a set with (potentially repeated) indices drawn from the multinomial distribution defined by $\tilde{w}_t^{1:N}$ and equal weights (of $\frac{1}{N}$). Resampling can be considered quantising the weights (i.e., introducing errors).[6] Since resampling introduces errors, it should only be employed when deemed necessary. This is typically achieved by monitoring the Effective Sample Size (ESS) [102] and only resampling when the ESS drops below a threshold (e.g., $\frac{N}{2}$). The ESS is defined as follows:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} \left(w_t^i\right)^2}. \tag{3.51}$$

The generic operation of a particle filter is then summarised in algorithm 3.

Algorithm 3 Generic particle filter [80].

Suppose we have $x_{t-1}^{(i)}$ and $w_{t-1}^{(i)}$ for $i = 1 \ldots N$ from the previous time step and the current measurement, z_t.

1: **for** Each particle i **do**
2: Sample particle from proposal: $x_t^{(i)} \sim q\left(x_t|x_{t-1}^{(i)}, z_t\right)$ using the prior or algorithms 4, 5 or 6.
3: Calculate new weights using (3.47).
4: **end for**
5: Normalise the weights using (3.50).
6: Calculate the effective sample size, N_{eff}, using (3.51).
7: **if** $N_{eff} < N_T$ **then**
8: Resample (by drawing the indices of the N new particles from the multinomial distribution defined by the vector $w_t^{(i)}$).
9: Set all weights to be $\frac{1}{N}$.
10: **end if**

$x_t^{(i)}$ and $w_t^{(i)}$ for $i = 1 \ldots N$ are then the new particles.

[6] We used a specific resampling algorithm, stratified sampling, in all implementations in this chapter (see, e.g., [80] for implementation details) since it minimises the size of these errors.

3.4.1 Bootstrap Particle Filter

Before considering the optimal proposal, it is worth highlighting that it is valid (and widespread in the context of Rao-Blackwellised particle filters) to use the prior as the proposal (as described in [80] or [94]) such that:

$$q\left(x_t|x_{t-1}, z_t\right) = p\left(x_t|x_{t-1}\right). \tag{3.52}$$

3.4.2 Optimal Proposal and near-Optimal Proposal for Particle Filter

This section will describe several (near-)optimal proposal methods and discuss the when they could be used using a few visualisations. An experiment will be conducted to show the differences of those methods with using a highly non-linear measurement model.

3.4.2.1 Description

Section 3.3.2 highlighted the importance of choosing a good proposal distribution in the context of importance sampling. Of course, there is a trade-off between choosing a proposal that is computationally expensive to use but generates useful samples and a proposal that is computationally inexpensive but does not generate such useful samples.

At one extreme, a very popular (and simple) choice of proposal is the prior $q\left(x_t|x_{t-1}, z_t\right) = p\left(x_t|x_{t-1}\right)$ [103, 104]. A particle filter that makes use of this proposal is often referred to as a bootstrap filter.

An alternative is to use an optimal proposal (more approaches are described in [84]):

$$q\left(x_t|x_{t-1}, z_t\right) = p\left(x_t|x_{t-1}, z_t\right) \propto p\left(x_t, z_t|x_{t-1}\right). \tag{3.53}$$

In contrast to the bootstrap filter, a particle filter that uses an optimal proposal considers the value of the measurement, z_t, when sampling values of x_t. If the prior, $p\left(x_t|x_{t-1}\right)$ is Gaussian and the likelihood, $p\left(z_t|x_t\right)$ is linear and Gaussian, then $p\left(x_t|x_{t-1}, z_t\right)$ is Gaussian with parameters that can be calculated using Kalman filter equations. Unfortunately, it is not always the case that $p\left(x_t|x_{t-1}, z_t\right)$ has a convenient parametric form with parameters that can be readily calculated. However, the approximations used in the EKF and UKF[7] can be employed such that a local (to each particle) linear approximation to $p\left(x_t|x_{t-1}, z_t\right)$ is calculated. Particle filters using the resulting near-optimal proposals with an EKF and UKF are described in Algorithms 4 and 5, respectively.

Instead of using a local linear approximation in the state-space, an alternative approach is to exploit an alternative coordinate frame where the linear approximation is less harsh [85]. This is achieved by considering a problem-specific (invertible) non-linear transform to the state

[7] When the UKF's approximations are used, the resulting algorithm is often known as an unscented particle filter [82].

Algorithm 4 Sampling method using the EKF [84].

Suppose we have $x^{(i)}_{t-1|t-1}$ from the previous time step, the current measurement, z_t, the process noise, Q, and the measurement noise, R, where $f(.)$ is the transition function for the state prediction and $h(.)$ is the transformation function from the state-space to measurement space.

1: Perform state propagation:

$$\mu_{t|t-1} = f(x^{(i)}_{t-1|t-1}) \tag{3.54}$$

2: Calculate:

$$\begin{aligned}\mu_{z,t|t-1} =& h\left(\mu_{t|t-1}\right)\\ \Sigma_{zz,t|t-1} =& H_t Q H_t^T + R\\ \Sigma_{xz,t|t-1} =& Q H_t^T\end{aligned}$$

where

$$H_t = \left.\frac{\partial h(.)}{\partial x}\right|_{\mu_{t|t-1}}$$

3: Calculate the parameters of the Gaussian approximation:

$$\begin{aligned}\mu_{t|t} =& \mu_{t|t-1} + \Sigma_{xz,t|t-1}{\Sigma_{zz,t|t-1}}^{-1}\left(z_t - \mu_{z,t|t-1}\right)\\ \Sigma_{t|t} =& Q - \Sigma_{xz,t|t-1}{\Sigma_{zz,t|t-1}}^{-1}{\Sigma_{xz,t|t-1}}^{T}\end{aligned}$$

4: Sample:

$$x^{(i)}_{t|t} \sim \mathcal{N}\left(\mu_{t|t}, \Sigma_{t|t}\right)$$

Then, $x^{(i)}_{t|t}$ is the sample from a near-optimal proposal that uses the EKF.

such that the measurement is a linear function of the transformed state, $r_t = r(x_t)$. The prior, $p(x_t|x_{t-1})$, is converted to a prior in the transformed space using the first derivative, $\frac{\partial r(x_t)}{\partial x_t}$, or unscented transform. Then, the Kalman filter equations are used to calculate a Gaussian approximation $q_r(r_t|x_{t-1}, z_t) \approx p(r_t|x_{t-1}, z_t)$. A sample, r^i_t, is then drawn from $q_r(r_t|x_{t-1}, z_t)$. Finally, the sampled value in the state-space is calculated using the inverse of the non-linear transform applied to $x^i_t = r^{-1}\left(r^i_t\right)$. Note that, when evaluating the proposal distribution, $q(x_t|x_{t-1}, z_t)$, the effect of the transformation needs to be considered, as follows:

$$q(x_t|x_{t-1}, z_t) = q_r\left(r(x_t)|x_{t-1}, z_t\right)\left|\left|\frac{\partial r(x_t)}{\partial x_t}\right|\right|, \tag{3.55}$$

where $\left|\left|\frac{\partial r(x_t)}{\partial x_t}\right|\right|$ is the absolute value of the determinant of the Jacobian matrix associated with the non-linear transform. This approach is described in Algorithm 6.

The aforementioned techniques for defining near-optimal proposals attempt to use judicious approximations but can give rise to poor performance in certain situations. As explained in Section 3.3.2, this can occur because the bound on the variance of the importance sampler is not finite (i.e., this can occur when the proposal is insufficiently heavy tailed). More specifically, since the EKF can underestimate the variance of a distribution, using the EKF approximation in

Algorithm 5 Sampling method using the UKF ([84] or [105]).

Suppose we have $x^{(i)}_{t-1|t-1}$ from the previous time step, the current measurement, z_t, the process noise, Q, and the measurement noise, R, where $f(.)$ is the transition function for state prediction and $h(.)$ is the transformation function from state-space to measurement space.

1: Perform state propagation:

$$\mu_{t|t-1} = f(x^{(i)}_{t-1|t-1})$$

2: Choose weighted sigma points ($\chi^{(j)}_{t|t-1}$ $j = 1, ..., 2D+1$):

$$\begin{aligned} \chi^{(j)}_{t|t-1} &= \mu_{t|t-1}, && j = 1 \\ \chi^{(j)}_{t|t-1} &= \mu_{t|t-1} \pm \left(\sqrt{(L+\lambda)Q}\right), && j = 2, ..., N \end{aligned}$$

The weights are calculated according to A.28.

3: Calculate

$$\begin{aligned} \mu_{z,t|t-1} &= \sum_{j=1}^{2D+1} W^{(j)}_{\mu,t|t-1} h\left(\chi^{(j)}_{t|t-1}\right) \\ \Sigma_{zz,t|t-1} &= \sum_{j=1}^{2D+1} W^{(j)}_{\sigma,t|t-1} \left[h\left(\chi^{(j)}_{t|t-1}\right) - \mu_{y,t|t-1}\right] \left[h\left(\chi^{(j)}_{t|t-1}\right) - \mu_{y,t|t-1}\right]^T + R \\ \Sigma_{xz,t|t-1} &= \sum_{j=1}^{2D+1} W^{(j)}_{\sigma,t|t-1} \left[\chi^{(j)}_{t|t-1} - \mu^{(i)}_x\right] \left[h\left(\chi^{(j)}_{t|t-1}\right) - \mu_{y,t|t-1}\right]^T \end{aligned}$$

4: Calculate the parameters of the Gaussian approximation:

$$\begin{aligned} \mu_{t|t} &= \mu_{t|t-1} + \Sigma_{xz,t|t-1}{\Sigma_{zz,t|t-1}}^{-1}\left(z_t - \mu_{z,t|t-1}\right) \\ \Sigma_{t|t} &= Q - \Sigma_{xz,t|t-1}{\Sigma_{zz,t|t-1}}^{-1}{\Sigma_{xz,t|t-1}}^T \end{aligned}$$

5: Sample:

$$x^{(i)}_{t|t} \sim \mathcal{N}\left(\mu_{t|t}, \Sigma_{t|t}\right)$$

Then, $x^{(i)}_{t|t}$ is the sample from a near-optimal proposal that uses the UKF.

a proposal can lead to inferior performance and should be approached with care. An unscented particle filter is a more robust alternative.

3.4.2.2 Discussion

To understand when the different proposals each have utility, it is helpful to begin with a scenario in which the likelihood is much more diffuse than the prior. The prior, likelihood, and near-optimal proposals are shown for such a scenario in Figure 3.1. It should be clear that the prior and near-optimal proposals are very similar. In such a scenario, it is unlikely that it is worthwhile to expend additional computational effort to draw samples from the near-optimal proposal rather than the prior.

Algorithm 6 Sampling method using a state-space transformation (EKF version) [85].

Suppose we have $x^{(i)}_{t-1|t-1}$ from previous time step, the current measurement, z_t, the process noise, Q, and the measurement noise R, where $f(.)$ is the transition function for state prediction, and $r\,(.)$ is the (non-linear and invertible) transformation function from state-space to measurement space such that H is the measurement matrix in the transformed space.

1: Perform state propagation: $\mu_{t|t-1} = f(x^{(i)}_{t-1|t-1})$.

2: Calculate:

$$\begin{aligned}\mu_{r,t|t-1} =& r\left(\mu_{t|t-1}\right)\\ \Sigma_{rr,t|t-1} =& G_t Q G_t^T\end{aligned}$$

where

$$G_t = \left.\frac{\partial r\,(.)}{\partial x}\right|_{\mu_{t|t-1}}$$

3: Calculate:

$$\begin{aligned}\Sigma_{zz,t|t-1} =& H\Sigma_{rr,t|t-1}H^T + R\\ \Sigma_{rz,t|t-1} =& \Sigma_{rr,t|t-1}H^T\end{aligned}$$

4: Calculate the parameters of the Gaussian approximation (in the transformed space):

$$\begin{aligned}\mu^{(i)}_{r,t|t} =& \mu_{r,t|t-1} + \Sigma_{rz,t|t-1}{\Sigma_{zz,t|t-1}}^{-1}\left(z_t - H\mu_{r,t|t-1}\right)\\ \Sigma^{(i)}_{t|t} =& \Sigma_{rr,t|t-1} - \Sigma_{rz,t|t-1}{\Sigma_{zz,t|t-1}}^{-1}{\Sigma_{rz,t|t-1}}^T\end{aligned}$$

5: Sample:

$$\begin{aligned}r^{(i)}_{t|t} \sim& \mathcal{N}\left(\mu^{(i)}_{r,t|t}, P^{(i)}_{t|t}\right)\\ x^{(i)}_{t|t} =& r^{-1}\left(r^{(i)}_{t|t}\right)\end{aligned}$$

Then, $x^{(i)}_{t|t}$ is the sample from a near-optimal proposal that uses a state-space transformation.

Conversely, there are scenarios in which the prior is more diffuse than the likelihood proposal. Such a scenario is shown in Figure 3.2. In this case, if samples were drawn from the prior, most of them will be placed in positions in which the likelihood is low. These samples will then have low weights and will contribute very little to the estimates. Indeed, when the prior is the proposal in such a scenario, only a small number of particles will contribute significantly to estimates. Conversely, it should be evident that the near-optimal proposal is similar to the likelihood. It is in this kind of scenario that the extra computational expense associated with using a near-optimal proposal could be preferable to simply drawing larger numbers of samples from the prior.

Figure 3.3 shows an exemplar non-linear likelihood and proposals based on using the approximations adopted by the EKF and UKF. It is evident that the EKF proposal is less diffuse than the UKF proposal and furthermore that the EKF proposal might be underestimating the uncertainty present (i.e., the EKF proposal might be insufficiently heavy tailed). Figure 3.3 also shows the proposal using a state-space transformation. It is evident that this proposal is

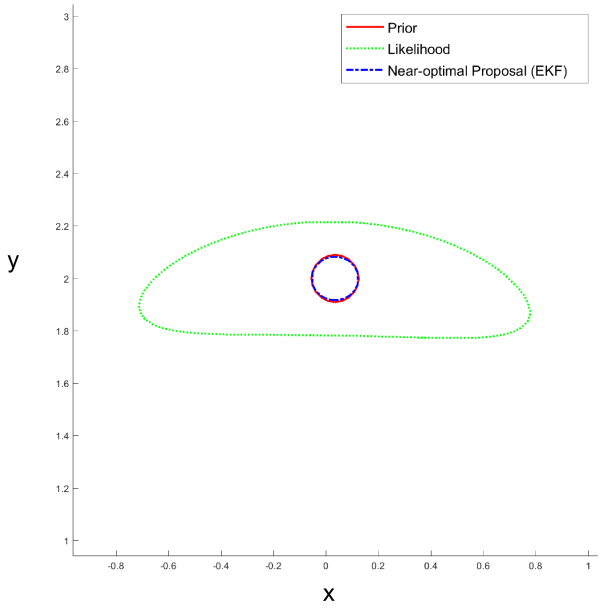


FIGURE 3.1: A visualisation of the probability density of the prior, likelihood, and near-optimal proposals, when the likelihood is more diffuse than the prior.

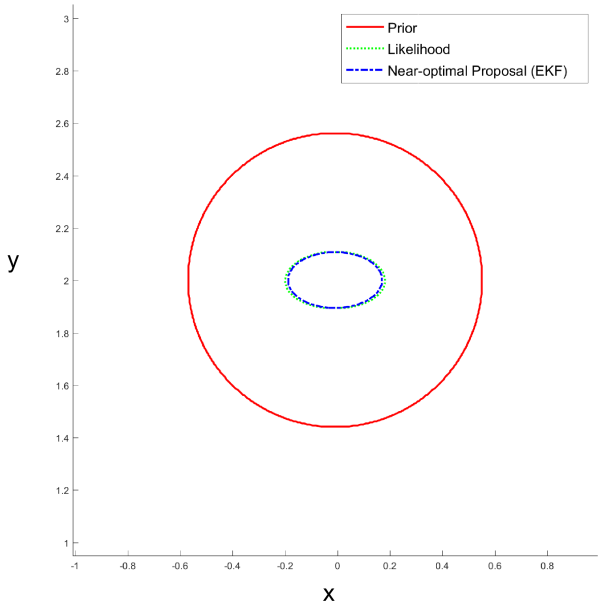


FIGURE 3.2: A visualisation of the probability density of the prior, likelihood, and near-optimal proposals, when the prior is more diffuse than the likelihood.

non-Gaussian (in the state-space) and appears to offer a close approximation to the likelihood. The effects of these observations on the results obtained using such proposals will be discussed in Section 3.4.5.

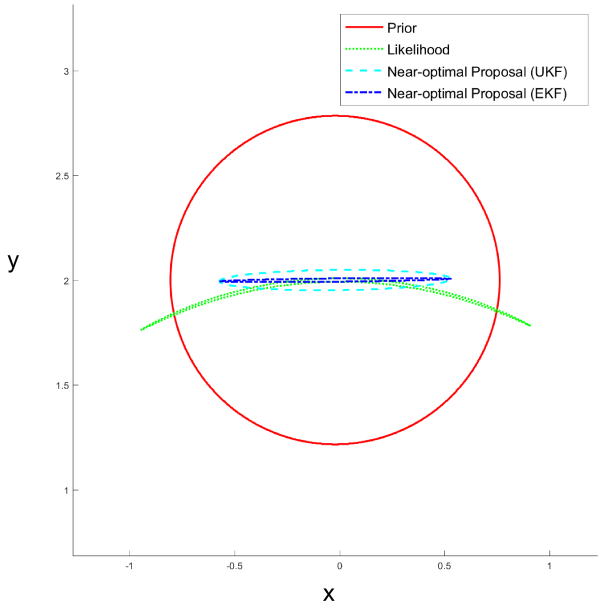
3.4.2.3 Experiment: Near-optimal Proposals Using EKF, UKF and State-space Transformation

While there are many papers comparing the relative performance of the UKF and EKF, there are fewer that discuss the differences when they are applied to generating a near-optimal proposal using the aforementioned form of process noise space in a particle filter. Moreover, no paper exists that takes the approach of sampling with state-space transformation [85] in the comparisons. The author takes the opportunity to briefly assess those differences here,[8] while also providing a baseline that aims to help the reader contextualise the results shown in the subsequent subsections.

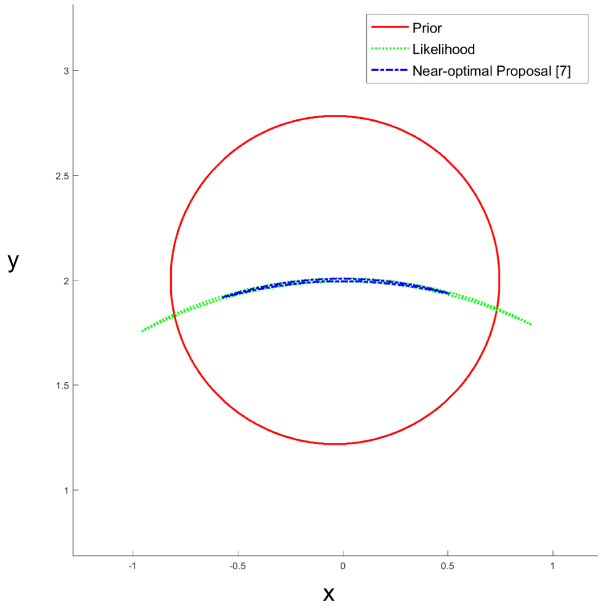
A brief introduction to the experiment is as follows. A near constant velocity model and two-dimensional radar measurement model are considered for single target tracking problem. The detailed variables of simulation are described in Section 3.4.5. In this section, we only consider a highly non-linear setting where the measurement noise, $\sigma_R^2 = 10^{-10}$ and $\sigma_\theta^2 = 10^{-6}$. The process noise intensity is $q_{int} = 0.3$. We assume the initial velocity uncertainty is $\sigma_V = 0.5$, which is considered small.

Figure 3.4 shows the root mean square error (RMSE) of those three methods when using 1000 particles. It is evident that the method that considers the state-space transformation generates a better estimation than the UKF proposal and that the UKF proposal is better than the EKF proposal at most epochs. For more general results, Table 3.1 shows the average RMSEs from the three methods using different numbers of particles. The ranking order of the performance is still anticipated as the case with 1000 particles. Comparing the UKF and EKF proposals, the performance of the UKF proposal with 1000 particles is similar to the performance using the EKF proposal with 5000 particles. As claimed by [83], although the computational cost of the UKF is not significantly heavier (when the dimensionality increases, more sigma points are needed, and the matrix inversions make UKF slower) than that of the EKF, Table 3.2 illustrates that the UKF is still computationally heavier. However, by considering accuracy and efficiency, we still infer that the UKF should be advocated. Regarding the state-space transformed proposal, it is even better than the UKF proposal, and the time complexity is the smallest (Table 3.2). The conclusion that the state-space transformed proposal is preferred over the UKF proposal, which is preferred over the EKF proposal, is obvious in some scenarios involving pronounced non-linearities such as the scenarios considered here.

[8] Reminding that the algorithm considered in this subsection does not involve any application of Rao-Blackwellisation.



(a) UKF and UKF proposal.



(b) State-space transformed [85] proposal.

FIGURE 3.3: A visualisation of the probability density of true likelihood, the UKF proposal, the EKF proposal, and the transformed proposal.

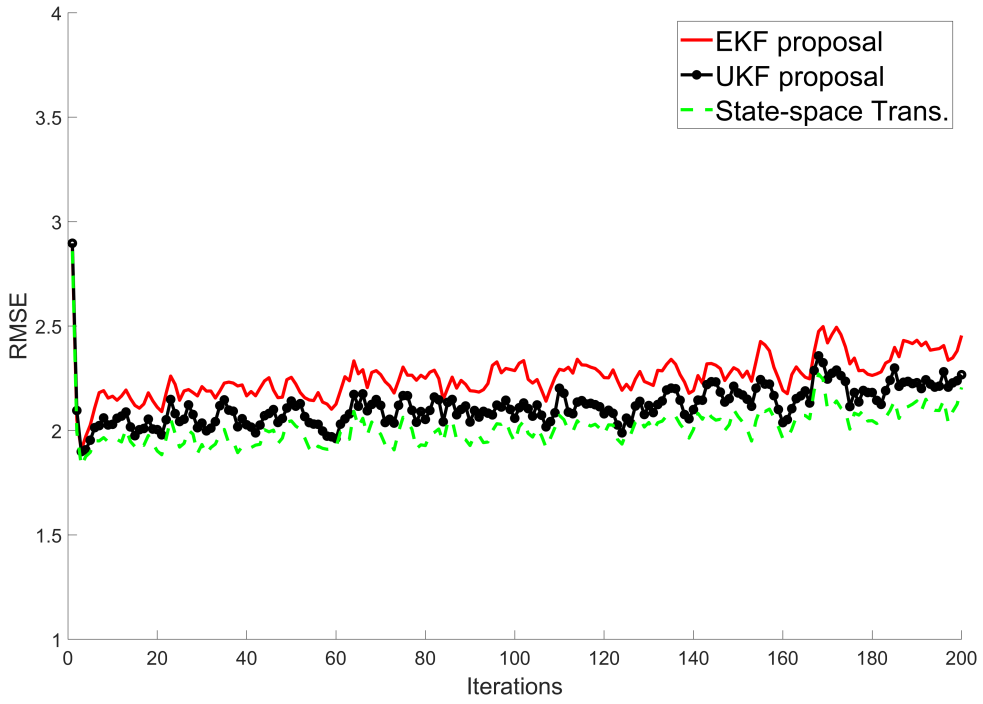


FIGURE 3.4: Position root mean square error (RMSE) for range-bearing tracking using 1500 particles.

N	EKF for proposal	UKF for proposal	State-space Trans. for proposal
100	4.19 ± 0.37	3.59 ± 0.20	2.30 ± 0.12
250	2.93 ± 0.19	2.58 ± 0.12	2.11 ± 0.10
500	2.52 ± 0.12	2.30 ± 0.11	2.05 ± 0.10
750	2.39 ± 0.11	2.21 ± 0.10	2.03 ± 0.10
1000	2.32 ± 0.11	2.17 ± 0.10	2.02 ± 0.09
1500	2.26 ± 0.10	2.12 ± 0.10	2.01 ± 0.09
3000	2.17 ± 0.10	2.07 ± 0.10	2.01 ± 0.09
5000	2.14 ± 0.10	2.04 ± 0.09	2.00 ± 0.09

TABLE 3.1: The results (averages and variations (after $\pm$ sign)) from the 1000 Monte-Carlo simulations comparing EKF and UKF for generating a near-optimal proposal.

N	EKF for proposal	UKF for proposal	State-space Trans. for proposal
100	2.09	3.38	2.35
250	5.23	8.49	5.93
500	10.49	16.99	11.79
750	15.95	25.63	17.77
1000	21.50	34.45	23.91
1500	32.77	52.22	36.16
3000	68.83	107.56	74.49
5000	123.09	187.69	129.46

TABLE 3.2: The average time taken in seconds for each simulation (200 iterations) from 1000 Monte-Carlo simulations as a function of the near-optimal proposal method, where N is the number of particles.

3.4.3 Rao-Blackwellised Particle Filter

This section provides a general description of Rao-Blackwellisation considering the correlated process noise (which has been described in [21,106], but does not appear to be widely adopted). Using this description as a basis, how to use an optimal proposal in conjunction with Rao-Blackwellisation will then be described.

The idea behind Rao-Blackwellisation (see, for example, [93]) is to decompose a large problem into two sub-problems: a linear sub-problem and a non-linear sub-problem. The linear sub-problem can then be tackled using a Kalman filter. In certain scenarios, the result of this decomposition will be a reduction in overall computational cost. To understand the approach, consider the case:

$$x_t = \begin{bmatrix} x_t^p \\ x_t^k \end{bmatrix}, \tag{3.56}$$

where the notation used identifies the parts of the problem to be tackled by the (p)article filter and the (k)alman[9] filter. The ith particle in the particle filter then comprises a sample, $x_{1:t}^{p,(i)}$, and the current mean, $\mu_{t|t}^{k,(i)}$, and covariance, $\Sigma_{t|t}^{k,(i)}$, output by a Kalman filter.

The (assumed linear) dynamics of the entire problem is defined as:

$$x_t = Fx_{t-1} + Q^{\frac{1}{2}}\omega_t, \tag{3.57}$$

[9]With posthumous apologies to Kalman for not capitalising his surname in the interests of the readability of this chapter.

which can be written as:

$$\begin{bmatrix} x_t^p \\ x_t^k \end{bmatrix} = \begin{bmatrix} F_{pp} & F_{pk} \\ F_{kp} & F_{kk} \end{bmatrix} \begin{bmatrix} x_{t-1}^p \\ x_{t-1}^k \end{bmatrix} + \begin{bmatrix} Q_{pp} & Q_{pk} \\ Q_{kp} & Q_{kk} \end{bmatrix}^{\frac{1}{2}} \begin{bmatrix} \omega_t^p \\ \omega_t^k \end{bmatrix}, \tag{3.58}$$

where the process noise follows a multivariate zero-mean Gaussian distribution.

We also assume that the measurement model can be written as:

$$z_t = h\left(x_t^p\right) + \epsilon_t. \tag{3.59}$$

Note that a more general case for the dynamics (3.57) and measurement model (3.59) can be found in [93].

3.4.3.1 Particle Filter Sub-problem

We consider the following target distribution for the particle filter [103]:

$$\begin{aligned} \pi\left(x_{1:t}^p\right) =& p\left(x_{1:t}^p, z_{1:t}\right) && (3.60)\\ =& p\left(x_{1:t}^p\right) p\left(z_{1:t}|x_{1:t}^p\right) && (3.61)\\ =& p\left(x_1^p\right) \prod_{t'=2}^{t} p\left(x_{t'}^p|x_{1:t'-1}^p\right) \prod_{t'=1}^{t} p\left(z_{t'}|x_{t'}^p\right) && (3.62)\\ =& p\left(x_{1:t-1}^p, z_{1:t-1}\right) p\left(x_t^p|x_{1:t-1}^p\right) p\left(z_t|x_t^p\right), && (3.63) \end{aligned}$$

where the dependence of the dynamics on $x_{1:t-2}^p$ contrasts with (3.44).

By inspecting (3.58), it is clear that:

$$p\left(x_t^p|x_{1:t-1}^p\right) = \mathcal{N}\left(x_t^p; \mu_{t|t-1}^p, \Sigma_{t|t-1}^p\right), \tag{3.64}$$

where

$$\begin{aligned} \mu_{t|t-1}^p =& F_{pp} x_{t-1}^{p,(i)} + F_{pk} \mu_{t-1}^k && (3.65)\\ \Sigma_{t|t-1}^p =& Q_{pp} + F_{pk} \Sigma_{t-1}^{k,(i)} F_{pk}^T. && (3.66) \end{aligned}$$

We consider the following structure of proposal distribution:

$$q\left(x_{1:t}^p|z_{1:t}\right) = q\left(x_t^p|x_{t-1}^p, x_{1:t-2}^p, z_t\right) q\left(x_{1:t-1}^p|z_{1:t-1}\right), \tag{3.67}$$

where, again, the inclusion of $x_{1:t-2}^p$, here in the incremental proposal, contrasts with (3.46). The proposal will be discussed in more detail in Section 3.4.4).

This gives rise to the following weight update equation:

$$w_t = w_{t-1} \frac{p\left(x_t^p | x_{t-1}^p, x_{1:t-2}^p\right) p\left(z_t | x_t^p\right)}{q\left(x_t^p | x_{t-1}^p, x_{1:t-2}^p, z_t\right)}. \tag{3.68}$$

3.4.3.2 Kalman Filter Sub-problem

As will be shown, the process noise involved in the dynamics for x_t^k is, in general, correlated with that involved in the dynamics of x_t^p. To understand the reason for the correlation, it is helpful to pick the following specific structure of the square root for Q:

$$\begin{bmatrix} Q_{pp} & Q_{pk} \\ Q_{kp} & Q_{kk} \end{bmatrix}^{\frac{1}{2}} = \begin{bmatrix} (Q_{pp} - Q_{pk} Q_{kk}^{-\frac{1}{2}^T} Q_{kk}^{-\frac{1}{2}} Q_{pk}^T)^{\frac{1}{2}} & Q_{pk} Q_{kk}^{-\frac{1}{2}^T} \\ 0 & Q_{kk}^{\frac{1}{2}} \end{bmatrix}. \tag{3.69}$$

Given this structure, we can write:

$$x_t^p = F_{pp} x_{t-1}^p + F_{pk} x_{t-1}^k + \left(Q_{pp} - Q_{pk} Q_{kk}^{-\frac{1}{2}^T} Q_{kk}^{-\frac{1}{2}} Q_{pk}^T\right)^{\frac{1}{2}} \omega_t^p + Q_{pk} Q_{kk}^{-\frac{1}{2}^T} \omega_t^k, \tag{3.70}$$

and

$$x_t^k = \underbrace{F_{kk}}_{\Phi} x_{t-1}^k + \underbrace{F_{kp} v_{t-1}^p}_{b_t} + \underbrace{Q_{kk}^{\frac{1}{2}}}_{\Gamma} \omega_t^k. \tag{3.71}$$

Moreover, (3.71) implies that:

$$x_{t-1}^k = F_{kk}^{-1} x_t^k - F_{kk}^{-1} F_{kp} x_{t-1}^p - F_{kk}^{-1} Q_{kk}^{\frac{1}{2}} \omega_t^k, \tag{3.72}$$

which together with (3.70) makes it possible to write:

$$\begin{aligned} &\underbrace{x_t^p + (F_{pk} F_{kk}^{-1} F_{kp} - F_{pp}) x_{t-1}^p}_{z_t} = \\ &\underbrace{F_{pk} F_{kk}^{-1}}_{G} x_t^k + \underbrace{\left(Q_{pp} - Q_{pk} Q_{kk}^{-\frac{1}{2}^T} Q_{kk}^{-\frac{1}{2}} Q_{pk}^T\right)^{\frac{1}{2}}}_{\Psi} \omega_t^p + \underbrace{\left(Q_{pk} Q_{kk}^{-\frac{1}{2}^T} - F_{pk} F_{kk}^{-1} Q_{kk}^{\frac{1}{2}}\right)}_{\Lambda} \omega_t^k, \end{aligned} \tag{3.73}$$

which makes it possible to describe the linear sub-problem as follows:

$$x_t^k = \Phi x_{t-1}^k + b_t + \Gamma \omega_t^k, \tag{3.74}$$

$$z_t = G x_t^k + \Psi \omega_t^p + \Lambda \omega_t^k. \tag{3.75}$$

The above equations then allow a Kalman filter to be implemented.[10]. Note that the measurement and process noise are correlated (i.e., $\Gamma \neq 0$) such that a slightly non-standard version needs to be used (see, e.g., [106], for example, for what we perceive to be an accessible derivation of this variant of the Kalman filter). This component of a particle filter is summarised in Algorithm 7.

Algorithm 7 Generic Rao-Blackwellisation in a particle filter.

Suppose we have, for each particle, $x_{t-1}^{p,(i)}$, $x_t^{p,(i)}$, $\mu_{t-1|t-1}^{k,(i)}$ and $P_{t-1|t-1}^{k,(i)}$. The parameters are F_{pp}, F_{pk}, F_{kp}, F_{kk}, Q_{pp}, Q_{pk}, Q_{kp}, and Q_{kk}.

1: Perform Kalman filter prediction

$$\mu_{t|t-1}^{k,(i)} = \Phi \mu_{t-1|t-1}^{k,(i)} + b_t \tag{3.76}$$
$$\Sigma_{t|t-1}^{k,(i)} = \Phi \Sigma_{t-1|t-1}^{k,(i)} \Phi^T + \Gamma\Gamma^T \tag{3.77}$$

where

$$\Phi = F_{kk} \tag{3.78}$$
$$b_t = F_{kp} x_{t-1}^p \tag{3.79}$$
$$\Gamma = Q_{kk}^{\frac{1}{2}} \tag{3.80}$$

2: Calculate

$$\Sigma_{xz} = \Sigma_{t|t-1}^{k,(i)} G^T + \Gamma\Lambda^T \tag{3.81}$$
$$\Sigma_{zz} = G\Sigma_{t|t-1}^{k,(i)} G^T + \Psi\Psi^T + \Lambda\Lambda^T + G\Lambda\Gamma^T + \Gamma\Lambda^T G^T \tag{3.82}$$

where

$$G = F_{pk} F_{kk}^{-1} \tag{3.83}$$
$$\Psi = \left(Q_{pp} - Q_{pk} Q_{kk}^{-\frac{1}{2}^T} Q_{kk}^{-\frac{1}{2}} Q_{pk}^T \right)^{\frac{1}{2}} \tag{3.84}$$
$$\Lambda = Q_{pk} Q_{kk}^{-\frac{1}{2}^T} - F_{pk} F_{kk}^{-1} Q_{kk}^{\frac{1}{2}} \tag{3.85}$$

3: Perform Kalman filter update

$$z_t = x_t^p + \left(F_{pk} F_{kk}^{-1} F_{kp} - F_{pp} \right) x_{t-1}^p \tag{3.86}$$
$$\mu_{t|t}^{k,(i)} = \mu_{t|t-1}^{k,(i)} + \Sigma_{xz} \Sigma_{zz}^{-1} \left(z_t - G x_{t|t-1}^{k,(i)} \right) \tag{3.87}$$
$$\Sigma_{t|t}^{k,(i)} = \Sigma_{t|t-1}^{k,(i)} + \Sigma_{xz} \Sigma_{zz}^{-1} \Sigma_{xz}^T \tag{3.88}$$

$\mu_{t|t}^{k,(i)}$ and $\Sigma_{t|t}^{k,(i)}$ are then the updated parameters for the Kalman filter

[10] This structure is such that the covariance matrices are not dependent on the particle index. As a result (and as first explained in [107]), the covariance terms (and Kalman gain, $P_{xz} {P_{zz}}^{-1}$) can be calculated once and used or (re)used by all particles. This can considerably reduce the computational effort demanded by using Rao-Blackwellisation considerably.

3.4.3.3 Discussion

To help explain when Rao-Blackwellisation is useful, we consider an example involving representing a Gaussian distribution in two dimensions, which we denote as X and Y. We consider three representations. In the first, we have samples of X and Y (Figure 3.5(a)). Using such samples, we can represent the distribution, $p(X)$, by simply projecting the samples onto the X-dimension. The resulting distribution is the set of delta functions shown in Figure 3.5(b).[11] It should be clear that the density of samples is highest near the mean (the true mean is equal to 3).

In the second representation, we consider samples of Y with Rao-Blackwellised Gaussian distributions for X (Figure 3.6(a)).[12] Using this representation, we can describe the distribution, $p(X)$, by projecting the Gaussian distribution onto the X-dimension and then summing their individual densities. The resulting distribution is shown in Figure 3.6(b). Note that Figure 3.6(b) is smooth, whereas Figure 3.5(b) is not.

Finally, we consider samples of Y with Gaussian distributions (Figure 3.7(a)) for X where the variance in X is smaller than in Figure 3.6. With this representation, we can again describe the distribution, $p(X)$, by projecting the Gaussian distribution onto the X-dimension and then summing their densities. The resulting distribution is shown in Figure 3.7(b). As should be evident, Figure 3.7(b) is much more similar to Figure 3.5(b) than Figure 3.6(b). With Gaussian distributions that have covariances that are small relative to the overall covariance of X, the (thin) Gaussian distributions behave similarly to delta functions.

3.4.4 Particle Filter using Optimal Proposal and Rao-Blackwellisation with Correlated Process Noise

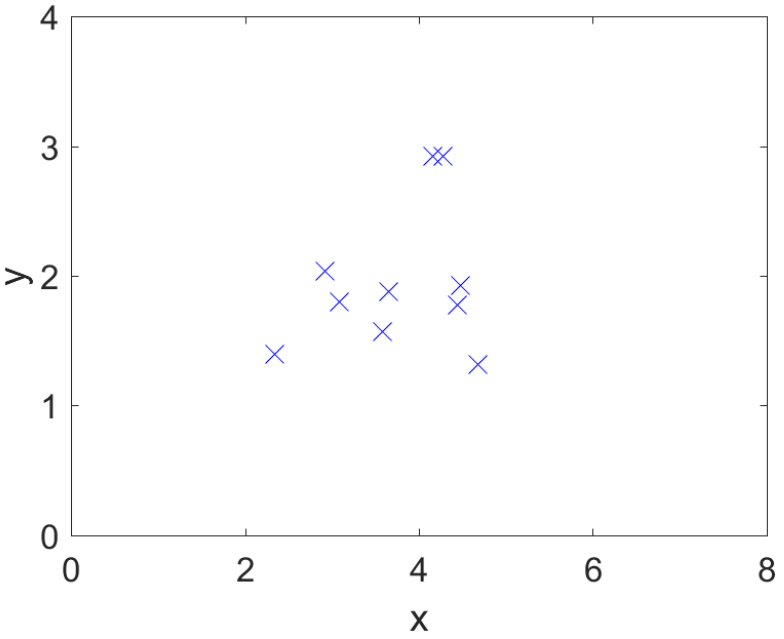
The implementation of Rao-Blackwellisation has been described above. This section will introduce how to obtain the near-optimal proposal. The combination of the two techniques will be described in the summary.

3.4.4.1 Near-Optimal Proposal

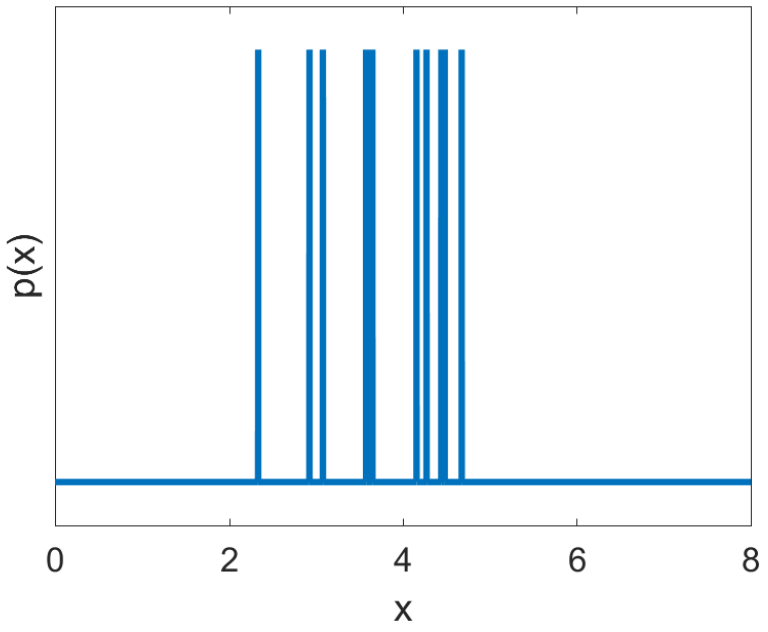
The near-optimal proposals (i.e., Algorithms 4, 5, and 6) have been described as accepting a prediction, μ_x and associated process noise covariance, Q as (some) inputs. As a result, these algorithms can be used in conjunction with any of the near-optimal proposals described in Section 3.4.2, but with $p\left(x_t^p | x_{1:t-1}^p\right)$ defining μ_x and Q. It is important to realise that such particle filters have two (separate) Kalman filters for each particle. One Kalman filter is used by the Rao-Blackwellisation. The other Kalman filter is used by the near-optimal proposal.

[11] The vertical axis of the figure is deliberately unlabelled since the delta functions should be infinitely thin and infinitely tall.

[12] The samples have been scaled such that the overall covariance in X and Y is unchanged.

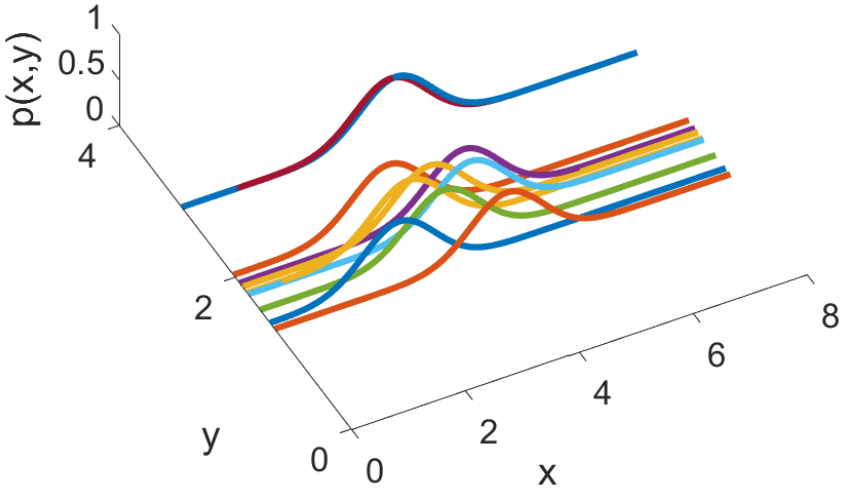


(a) In order to represent the objective distribution, we draw samples of X and Y.

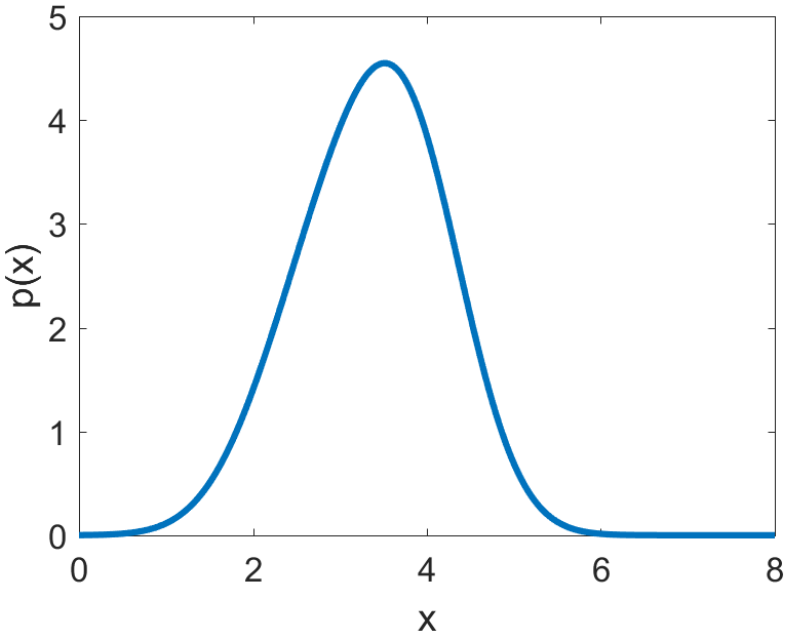


(b) We can approximate to $p(X)$ by the samples, note that the delta function is infinitely thin and tall.

FIGURE 3.5: Diagrammatic representation of 10 samples in X and Y with associated representation of $p(X)$.

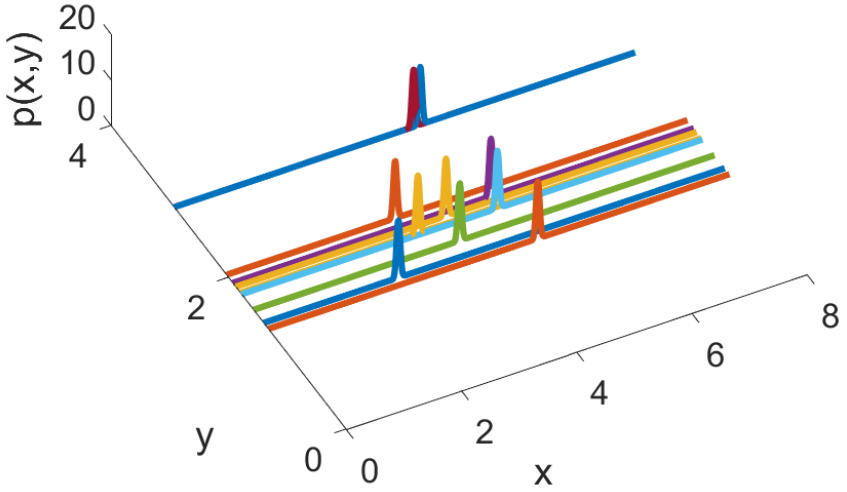


(a) In order to represent the objective distribution, we draw samples of Y with Gaussian for X (with a large variance).

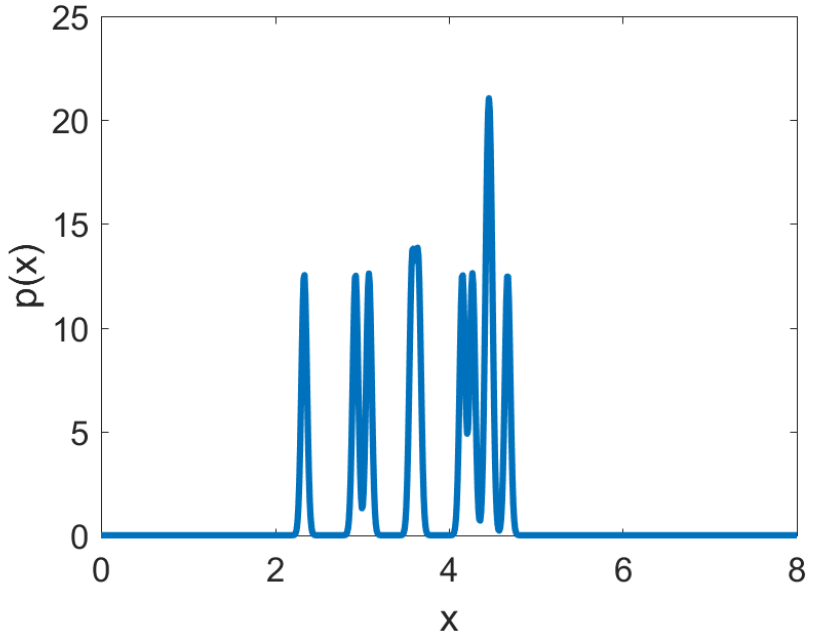


(b) The distribution, $p(X)$, can be described by summing the densities of all the samples.

FIGURE 3.6: Diagrammatic representation of 10 Rao-Blackwellised distributions with associated representation of $p(X)$. Large variance case.



(a) In order to represent the objective distribution, we draw samples of Y with Gaussian for X (with a small variance).



(b) The distribution, $p(X)$, can be described by summing the densities of all the samples.

FIGURE 3.7: Diagrammatic representation of 10 Rao-Blackwellised distributions with associated representation of $p(X)$. Small variance case.

3.4.4.2 Summary

Algorithm 8 describes a generic implementation of a Rao-Blackwellised particle filter.

Algorithm 8 Generic Rao-Blackwellised particle filter.

Suppose we have $x_{t-1}^{p,(i)}$, $\mu_{t-1|t-1}^{k,(i)}$, $\Sigma_{t-1|t-1}^{k,(i)}$ and $w_{t-1}^{(i)}$ for $i = 1 \dots N$ from the previous time step and the current measurement, z_t.

1: **for** Each particle i **do**
2: Sample particle from prior as Section 3.4.1 or $x_t^{p,(i)} \sim q\left(x_t^p | x_{t-1}^{p,(i)}, x_{1:t-2}^{p,(i)}, z_t\right)$ using the approaches in Section 3.4.4.1.
3: Calculate $\mu_{t|t}^{k,(i)}$ and $\Sigma_{t|t}^{k,(i)}$ using Algorithm 7.
4: Calculate new weights using (3.68).
5: **end for**
6: Normalise the weights using (3.50).
7: Calculate the effective sample size, N_{eff}, using (3.51).
8: **if** $N_{eff} < N_T$ **then**
9: Resample (by drawing the indices of the N new particles from the multinomial distribution defined by vector of $w_t^{(i)}$).
10: Set all weights to $\frac{1}{N}$.
11: **end if**

Then, $x_t^{p,(i)}$, $\mu_{t|t}^{k,(i)}$, $\Sigma_{t|t}^{k,(i)}$, and $w_t^{p,(i)}$ for $i = 1 \dots N$ are the new particles.

3.4.5 Experiments

The generic scenario we consider involves a single target moving with nearly constant velocity being detected by a (stationary) radar sensor (which is widely used, e.g., [85, 90]). The state vector of the target is $\left[X, Y, \dot{X}, \dot{Y}\right]^T$ with states corresponding to the two-dimensional Cartesian position and velocity of the target relative to the radar. The radar is assumed to generate (polar, so non-linear) measurements of range and bearing, $[R, \theta]^T$. The initial state of the target is $[2000, 2000, 0.7, 0.4]^T$.

The dynamic model described in (3.37) is adopted with:

$$F = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.89}$$

and the process noise covariance (introduced in [108]):

$$Q = \begin{bmatrix} \frac{1}{3}T^3 & 0 & \frac{1}{2}T^2 & 0 \\ 0 & \frac{1}{3}T^3 & 0 & \frac{1}{2}T^2 \\ \frac{1}{2}T^2 & 0 & T & 0 \\ 0 & \frac{1}{2}T^2 & 0 & T \end{bmatrix} q_{int}, \tag{3.90}$$

where q_{int} is the process noise intensity, and we assume that $T = 1$.

To initialise the particles, a prior, x_0 and P_0, is manually configured as in (3.91) and (3.92) for all the following experiments. For the non-Rao-Blackwellisation algorithm, the samples with both position and velocity are drawn from a near-optimal proposal which calculated by an EKF with considering the state-space transformation (recall Section 3.4.2). The near-optimal proposal uses the information of prior and the first measurement. Importance sampling is then used to define the initial particle weights. For the Rao-Blackwellisation algorithm, the samples only include positions. They are also drawn from near-optimal proposal, but without consider the velocities. The velocity part of the prior is directly use for initialising the Kalman filter in Rao-Blackwellisation.

$$x_0 = \begin{bmatrix} 2020 & 2020 & 0 & 0 \end{bmatrix}^T, \tag{3.91}$$

Reminding that this initialisation is different from the true initial state of the target.

$$P_0 = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & \sigma_V & 0 \\ 0 & 0 & 0 & \sigma_V \end{bmatrix}^2, \tag{3.92}$$

where σ_V is the uncertainty of the initial velocity which will be specified in each following scenario.

In Section 3.4.2.3, we have compared the use of the near-optimal proposal fulfilled by an UKF, an EKF, and using state-space transformation. In the subsequent subsections, we consider scenarios that aim to clarify when the use of a near-optimal proposal computed by state-space transformation (in the result that it is the best) and Rao-Blackwellisation are beneficial in isolation or combination. These scenarios differ in terms of the magnitude of process noise and initial velocity uncertainty. We also vary the range measurement noise σ_R, which is a portion of the measurement noise. Increasing the uncertainty in the range while keeping the uncertainty in the bearing constant has the effect of making the likelihood appear more non-linear. Rather than presenting all combinations, we present a subset that is sufficient to understand the relative merits of near-optimal proposals and Rao-Blackwellisation.

In the subsequent experiments, we consider 1000 Monte-Carlo runs of each scenario. Each Monte-Carlo run consists of 200 time steps and has a different target trajectory based on the process noise and different realisation of measurement noise. The results include the average RMSE and the variation in Euclidean distance between the estimation and the true position. In order to remove the influence from the very wrong estimations, the errors are sorted and over the range of the second to ninety-eighth percentile are included. The average RMSE as a function of t is also visualised. In Sections 3.4.5.2 and 3.4.5.3, the number of effective samples

(as calculated using (3.51)) is also reported.

In the following tables and figures, RB is the abbreviation for Rao-Blackwellisation, n-OP is the abbreviation for near-optimal proposal and 'No-' means without the technique.

3.4.5.1 Large Measurement Noise, Small Process Noise, and Good Velocity Initialisation

We now consider a scenario with significant uncertainty in the measurements, where $\sigma_R^2 = 1$ and $\sigma_\theta^2 = 10^{-8}$. We consider a small process noise intensity, $q_{int} = 10^{-2}$, and good initialisation of velocity with $\sigma_V = 0.5$. We compare the performance achieved using neither, one, and both of a near-optimal proposal and Rao-Blackwellisation.

Figure 3.8 shows the average RMSE as a function of time when using 1500 particles. From this figure, all four approaches achieve similar results at all time steps. The improvement from using either the near-optimal proposal or Rao-Blackwellisation is not significant. Table 3.3 shows how the results vary as the number of particles varies. When the number of particles is 100, using a near-optimal proposal, Rao-Blackwellisation or both does offer some improvement over using neither. However, a commensurate improvement can be achieved by a relatively slight increase in the number of particles. When the particle number is larger than 1000, all four approaches perform similarly.

Table 3.4 shows the average time consumed for one simulation. The implementations share common functions to try to ensure that the computational comparison is fair. Although the relative computational cost might change depending on different coding optimisations, we observe that the particle filters using Rao-Blackwellisation and the optimal proposal have an approximating computational cost to one another and that the cost is 3.5 times more than using neither. Moreover, using both Rao-Blackwellisation and the optimal proposal is the slowest method and is approximately 5 times slower than using neither. This is such that (for this example) the time taken to use neither technique with 3000 particles is similar to the time taken using one of the two with nearly 750 particles, which is similar to using both with 500 particles or so. Note that this summary is relevant to the results shown in Sections 3.4.5.1 to 3.4.5.3 since only the parameters are changed between those sections.

In this situation, considering the increase in computational load associated with using either a near-optimal proposal or Rao-Blackwellisation, we infer that a simple particle filter with a larger number of particles should be advocated.

3.4.5.2 Small Measurement Noise, Large Process Noise, and Good Velocity Initialisation

We next consider a scenario where the measurement noise is small, where $\sigma_R^2 = 10^{-4}$ and $\sigma_\theta^2 = 10^{-8}$. We consider a large process noise, $q_{int} = 0.1$ and good initialisation of velocity

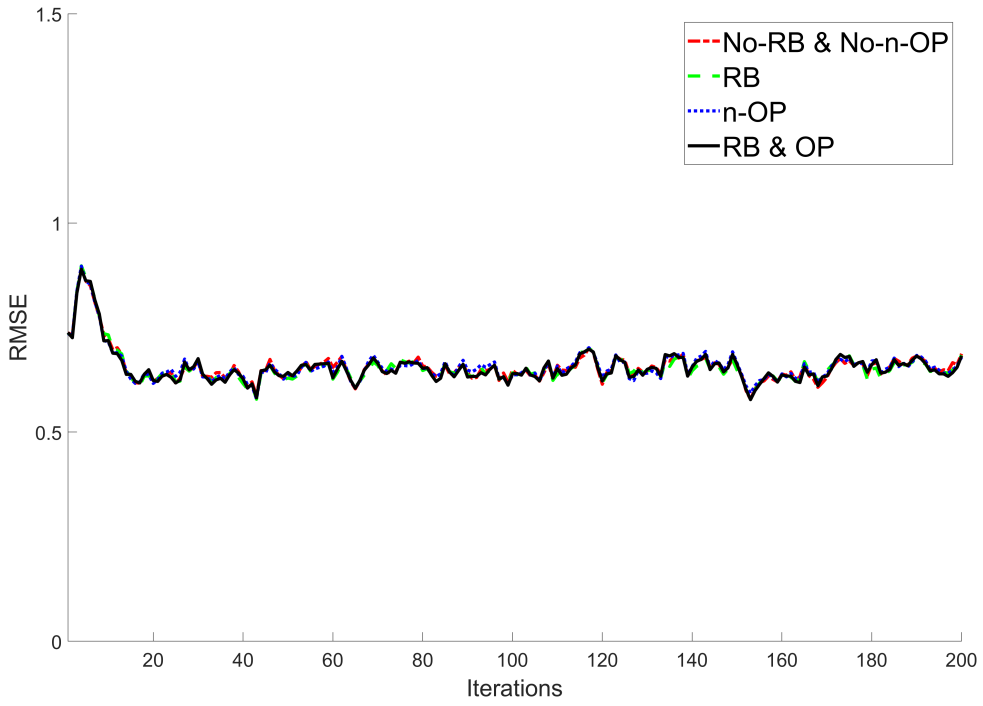


FIGURE 3.8: Average root mean square error (RMSE) graph as a function of time for scenario considered in Section 3.4.5.1.

N	No-RB&No-n-OP	RB	n-OP	RB&n-OP
100	1.00 ± 0.17	0.90 ± 0.12	0.93 ± 0.15	0.84 ± 0.10
250	0.75 ± 0.07	0.72 ± 0.07	0.74 ± 0.07	0.71 ± 0.06
500	0.69 ± 0.05	0.68 ± 0.05	0.69 ± 0.05	0.68 ± 0.05
750	0.67 ± 0.04	0.67 ± 0.04	0.67 ± 0.04	0.66 ± 0.04
1000	0.66 ± 0.04	0.66 ± 0.04	0.66 ± 0.04	0.66 ± 0.04
1500	0.66 ± 0.04	0.65 ± 0.04	0.66 ± 0.04	0.65 ± 0.04
3000	0.65 ± 0.04	0.65 ± 0.04	0.65 ± 0.04	0.65 ± 0.04
5000	0.65 ± 0.04	0.65 ± 0.04	0.65 ± 0.04	0.65 ± 0.04

TABLE 3.3: Average root mean square error (RMSE) and variations (after $\pm$ sign) for the scenario considered in Section 3.4.5.1 as a function of the algorithm and number of particles, N.

N	No-RB&No-n-OP	RB	n-OP	RB&n-OP
100	0.77	3.19	3.42	4.61
250	1.87	7.69	8.42	11.21
500	3.77	15.29	16.83	22.36
750	5.75	23.08	25.36	33.63
1000	7.80	30.88	33.85	44.84
1500	11.84	45.62	50.11	66.02
3000	26.80	95.71	104.24	136.97
5000	50.12	164.03	178.07	232.14

TABLE 3.4: Average time taken in seconds for each simulation (200 iterations) as a function of the algorithm and number of particles, N.

with $\sigma_V = 0.5$. Again, we compare the performance achieved using neither, one, and both of a near-optimal proposal and Rao-Blackwellisation.

From Table 3.5, we can see a clear benefit from using a near-optimal proposal in this scenario. The use of Rao-Blackwellisation does offer some improvement relative to not doing so, but the benefit is dwarfed by that associated with using a near-optimal proposal. When using both, a near-optimal proposal offers the best performance (only when using a very small number of particles smaller than 100), there is only a negligible improvement resulting from adding Rao-Blackwellisation to a particle filter using a near-optimal proposal. This is also clear from Figure 3.9, which shows how the RMSE of the approaches varies over time when using the different techniques or not with 1500 particles. Note that, in contrast to the scenario considered in Section 3.4.5.1, it is clear that a vast increase to the number of particles considered would be needed to enable a particle filter that does not use a near-optimal proposal to achieve the performance associated with a particle filter that does use a near-optimal proposal. Table 3.6 shows the number of effective samples, N_{eff}, for the same scenario. It is clear that the improvement in RMSE shown in Table 3.5 is mirrored by improvements in N_{eff}, but it is also clear that, without the near-optimal proposal, the effective sample size is a very small fraction of the total number of samples.

In this situation, the utility of a near-optimal proposal is clear, but considering the computational expense of also using Rao-Blackwellisation, it is unlikely that Rao-Blackwellisation would be advocated. Using neither technique is fast but it is acceptable only when 5000 particles are used. It can have a similar performance using only the near-optimal proposal with 100 particles, which is much faster according to Table 3.4.

3.4.5.3 Small Measurement Noise, Small Process Noise, and Poor Velocity Initialisation

Next, we consider a scenario with small measurement noise such that $\sigma_R^2 = 10^{-4}$ and $\sigma_\theta^2 = 10^{-8}$. We consider a small process noise intensity, $q_{int} = 10^{-2}$ and poor initialisation of velocity with $\sigma_V = 10$.

N	No-RB&No-n-OP	RB	n-OP	RB&n-OP
100	251.43 ± 214.47	38.66 ± 38.22	0.27 ± 0.01	0.26 ± 0.01
250	110.49 ± 100.74	0.66 ± 0.06	0.26 ± 0.01	0.26 ± 0.01
500	25.74 ± 25.52	0.41 ± 0.03	0.26 ± 0.01	0.26 ± 0.01
750	11.51 ± 13.89	0.35 ± 0.02	0.26 ± 0.01	0.26 ± 0.01
1000	1.11 ± 1.49	0.31 ± 0.02	0.26 ± 0.01	0.26 ± 0.01
1500	0.37 ± 0.03	0.29 ± 0.01	0.26 ± 0.01	0.26 ± 0.01
3000	0.30 ± 0.02	0.27 ± 0.01	0.26 ± 0.01	0.26 ± 0.01
5000	0.28 ± 0.01	0.26 ± 0.01	0.26 ± 0.01	0.26 ± 0.01

TABLE 3.5: Average root mean square error (RMSE) and variations (after $\pm$ sign) for the scenario considered in Section 3.4.5.2 as a function of the algorithm and number of particles, N.

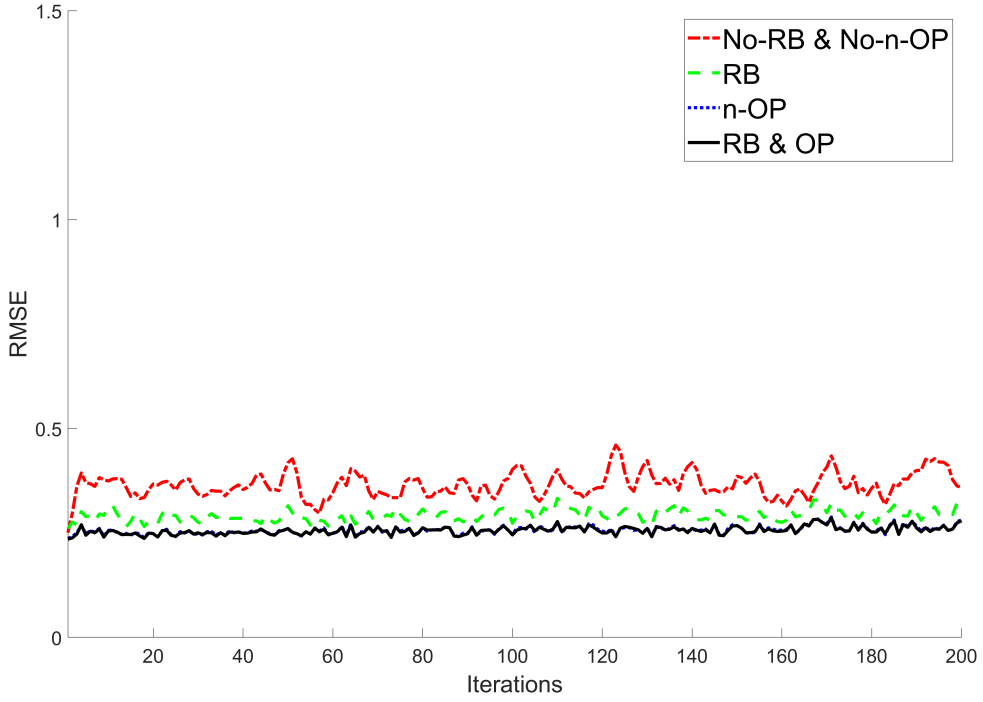


FIGURE 3.9: Average root mean square error (RMSE) graph as a function of time for the scenario considered in Section 3.4.5.2.

N	No-RB&No-n-OP	RB	n-OP	RB&n-OP
100	0.66	2.06	35.70	50.68
250	3.43	4.89	86.46	125.59
500	8.51	9.47	170.77	250.47
750	13.28	14.03	255.02	375.28
1000	18.03	18.63	339.56	499.91
1500	27.49	27.86	508.02	749.88
3000	55.30	55.53	1013.09	1498.41
5000	92.11	92.40	1686.92	2497.66

TABLE 3.6: Average N_{eff} as a function of algorithm and number of particles for the scenario considered in Section 3.4.5.2, including the average percentage of N (in the final row).

Inspection of the RMSE results shown in Table 3.7 (and the number of effective samples shown in Table 3.9) indicates that the benefits in this scenario of using near-optimal proposal are significant. Closer inspection of the RMSE errors at early time steps (Table 3.8) makes it clear that, while only comparing using near-optimal proposal and using both, the differences resulting from adding Rao-Blackwellisation are more pronounced at the early time steps than they are when averaged over all time steps. Indeed, in the context of poor initialisation, the benefit of using both rather than the one is also particularly apparent in Figure 3.10. It is evident that, when the near-optimal proposal is used, Rao-Blackwellisation improves the performance in the early iterations of the scenario but also that the benefit of using Rao-Blackwellisation then become less pronounced as time progresses. The reason for why only using near-optimal proposal is less effective could be that, with a very uncertain velocity initialisation, a large part of particles have wrong velocities in early iterations, and a small process noise makes it harder and takes longer to estimate the good velocities. But when increasing the number of particles, the number of effective velocities goes up. Using near-optimal proposal can yield same performance. The basic contribution of Rao-Blackwellisation is to provide a more efficient way to estimate the velocity. Besides, both using neither techniques and using only Rao-Blackwellisation are far from satisfactory. It is because the initialisation based on a large uncertain velocity generates few good velocity samples. After time propagation, most position estimation are away from the true and thus leads to a low effective sample size. It can be imagined that increasing the particle number can fix the issue, but the computational cost will definitely be compromised and exceeds the timetable for this thesis.

In this situation, the combination of Rao-Blackwellisation and a near-optimal proposal offers improvements over using either alone. In terms of computational cost, using both techniques with 100 particles can perform similarly to using the near-optimal proposal alone with 3000 particles. The particle filter using both techniques is advocated in this scenario.

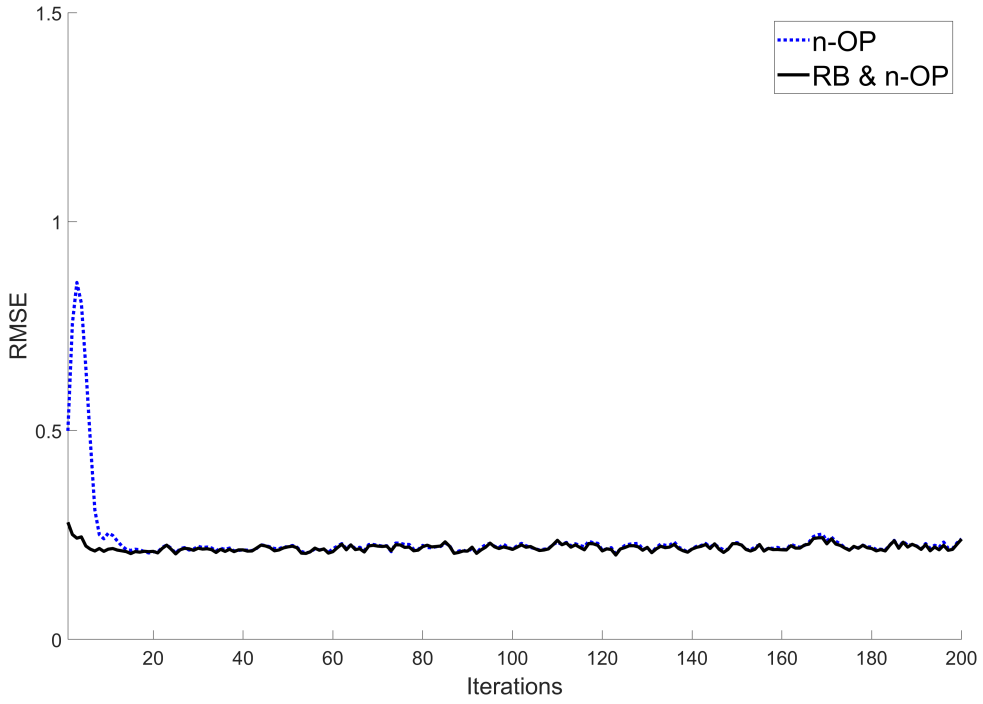


FIGURE 3.10: Average root mean square error (RMSE) graph as a function of time for the scenario considered in Section 3.4.5.3.

N	No-RB&No-n-OP	RB	n-OP	RB&n-OP
100	591.25 ± 342.40	576.22 ± 335.08	0.75 ± 1.22	0.23 ± 0.01
250	300.84 ± 181.38	308.79 ± 180.62	0.40 ± 0.58	0.22 ± 0.01
500	212.23 ± 125.84	182.69 ± 109.10	0.30 ± 0.31	0.22 ± 0.01
750	154.60 ± 92.59	144.76 ± 86.06	0.28 ± 0.23	0.22 ± 0.01
1000	124.70 ± 75.73	83.58 ± 50.11	0.26 ± 0.16	0.22 ± 0.01
1500	77.81 ± 48.77	59.79 ± 37.55	0.24 ± 0.08	0.22 ± 0.01
3000	10.65 ± 5.54	0.39 ± 0.50	0.23 ± 0.04	0.22 ± 0.01
5000	0.34 ± 0.35	0.29 ± 0.25	0.22 ± 0.02	0.22 ± 0.01

TABLE 3.7: Average root mean square error (RMSE) and variations (after $\pm$ sign) for the scenario considered in Section 3.4.5.3 as a function of the algorithm and number of particles, N.

Time Step #	No-RB&No-n-OP	RB	n-OP	RB&n-OP
5	5.20	4.41	0.65	0.22
10	8.12	6.18	0.25	0.22
15	10.94	8.08	0.21	0.21
20	14.29	10.62	0.21	0.21
25	17.69	13.20	0.20	0.20

TABLE 3.8: Average root mean square error (RMSE) for the first 20 time steps of the scenario considered in Section 3.4.5.3 as a function of the algorithm.

N	No-RB&No-n-OP	RB	n-OP	RB&n-OP
100	1.01	1.26	38.30	51.16
250	5.98	7.79	94.32	126.52
500	18.19	21.83	188.05	252.21
750	33.65	38.35	281.57	377.57
1000	52.32	60.09	375.60	503.25
1500	88.83	97.35	564.37	754.11
3000	208.80	212.80	1127.37	1507.46
5000	360.54	364.25	1878.74	2511.17

TABLE 3.9: Average N_{eff} as a function of the algorithm and number of particles for the scenario considered in Section 3.4.5.3, including the average percentage of N (in the final row).

3.4.5.4 Higher-Dimensional Simulations

To test our system with states of higher dimensionality, we considered a three dimensional radar measurement case as follows. The dynamic model which refers to Section 3.2.1.2:

$$F = \begin{bmatrix} 1 & 0 & 0 & T & 0 & 0 \\ 0 & 1 & 0 & 0 & T & 0 \\ 0 & 0 & 1 & 0 & 0 & T \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.93}$$

and

$$Q = \begin{bmatrix} \frac{1}{3}T^3 & 0 & 0 & \frac{1}{2}T^2 & 0 & 0 \\ 0 & \frac{1}{3}T^3 & 0 & 0 & \frac{1}{2}T^2 & 0 \\ 0 & 0 & \frac{1}{3}T^3 & 0 & 0 & \frac{1}{2}T^2 \\ \frac{1}{2}T^2 & 0 & 0 & T & 0 & 0 \\ 0 & \frac{1}{2}T^2 & 0 & 0 & T & 0 \\ 0 & 0 & \frac{1}{2}T^2 & 0 & 0 & T \end{bmatrix} q_{int}. \tag{3.94}$$

The measurement model is extended from Section 3.2.2.2:

$$h\left(x_t\right) = \begin{bmatrix} \sqrt{X_t^2 + Y_t^2 + Z_t^2} \\ arctan(Y_t, X_t) \\ arctan(Z_t, \sqrt{X_t^2 + Y_t^2}) \end{bmatrix}, \tag{3.95}$$

and

$$R = \begin{bmatrix} \sigma_R^2 & 0 & 0 \\ 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & \sigma_\phi^2 \end{bmatrix}. \tag{3.96}$$

All three scenarios that are similar to those associated with Sections 3.4.5.1, 3.4.5.2, and 3.4.5.3 are tested in this section, and with $\sigma_\theta^2 = \sigma_\phi^2$. All the results are presented in Tables 3.10, 3.11, and 3.12, respectively. From the results, it is clear that extending to a higher dimensionality demands more particles and that the general conclusions are not changed: 1) there is little benefit from using either Rao-Blackwellisation or a near-optimal proposal (or both) when the measurement noise is large, the process noise is small, and the velocity is well initialised. 2) when the measurement noise is small and process noise is large, applying near-optimal proposal can improve the performance huge; 3) When the initialisation is poor, the process noise and the measurement noise are small using both improves the performance relative to either alone.

N	No-RB&No-n-OP	RB	n-OP	RB&n-OP
100	1.78 ± 0.40	1.35 ± 0.36	1.38 ± 0.36	1.05 ± 0.23
250	0.97 ± 0.22	0.85 ± 0.19	0.87 ± 0.17	0.77 ± 0.14
500	0.76 ± 0.14	0.70 ± 0.11	0.72 ± 0.12	0.67 ± 0.09
750	0.69 ± 0.11	0.66 ± 0.09	0.68 ± 0.10	0.64 ± 0.07
1000	0.66 ± 0.10	0.63 ± 0.08	0.65 ± 0.09	0.62 ± 0.07
1500	0.63 ± 0.07	0.61 ± 0.07	0.62 ± 0.07	0.60 ± 0.06
3000	0.60 ± 0.05	0.59 ± 0.05	0.59 ± 0.05	0.59 ± 0.05
5000	0.58 ± 0.05	0.58 ± 0.04	0.58 ± 0.04	0.58 ± 0.04

TABLE 3.10: Average root mean square error (RMSE) and variations (after $\pm$ sign) for the scenario (large measurement noise, small process noise, and good velocity initialisation) described in Section 3.4.5.4 as a function of the algorithm and number of particles.

N	No-RB&No-n-OP	RB	n-OP	RB&n-OP
100	294.22 ± 243.82	116.29 ± 108.79	0.37 ± 0.01	0.33 ± 0.01
250	192.52 ± 169.96	22.40 ± 21.42	0.34 ± 0.01	0.32 ± 0.01
500	101.97 ± 92.15	0.80 ± 0.06	0.33 ± 0.01	0.32 ± 0.01
750	70.36 ± 65.02	0.60 ± 0.03	0.32 ± 0.01	0.32 ± 0.01
1000	25.54 ± 23.52	0.52 ± 0.03	0.32 ± 0.01	0.32 ± 0.01
1500	9.00 ± 10.44	0.45 ± 0.03	0.32 ± 0.01	0.31 ± 0.01
3000	0.50 ± 0.04	0.37 ± 0.01	0.32 ± 0.01	0.31 ± 0.01
5000	0.42 ± 0.02	0.35 ± 0.01	0.32 ± 0.01	0.31 ± 0.01

TABLE 3.11: Average root mean square error (RMSE) and variations (after $\pm$ sign) for the scenario (small measurement noise, large process noise, and good velocity initialisation) described in Section 3.4.5.4 as a function of the algorithm and number of particles.

N	No-RB&No-n-OP	RB	n-OP	RB&n-OP
100	828.95 ± 479.92	828.15 ± 480.36	2.09 ± 2.83	0.30 ± 0.01
250	521.39 ± 305.33	507.47 ± 294.54	1.15 ± 1.73	0.28 ± 0.01
500	392.13 ± 231.94	385.13 ± 226.47	0.81 ± 1.25	0.27 ± 0.01
750	317.63 ± 188.15	332.29 ± 197.37	0.64 ± 0.95	0.27 ± 0.01
1000	295.84 ± 177.70	283.23 ± 168.25	0.58 ± 0.86	0.27 ± 0.01
1500	244.80 ± 147.36	245.72 ± 148.51	0.47 ± 0.63	0.27 ± 0.01
3000	174.64 ± 106.96	165.77 ± 99.78	0.39 ± 0.45	0.27 ± 0.01
5000	129.06 ± 80.19	125.83 ± 78.62	0.34 ± 0.31	0.27 ± 0.01

TABLE 3.12: Average root mean square error (RMSE) and variations (after $\pm$ sign) for the scenario (small measurement noise, small process noise, and bad velocity initialisation) described in Section 3.4.5.4 as a function of the algorithm and number of particles.

3.4.5.5 Discussion

This section has considered variants of a scenario with the aim of understanding when the near-optimal proposal and Rao-Blackwellisation have utility individually and in combination. The results in Section 3.4.5.1 indicate that there are scenarios in which the uncertainty in the predicted state is small such that the measurements are not very informative relative to the dynamic prior and initialisation. In such a setting, using either or both methods is unlikely to have utility. However, the results in Section 3.4.5.2 highlight that this situation changes when the predicted state is highly uncertain, and the measurement is therefore highly informative. In that setting, using both a near-optimal proposal and Rao-Blackwellisation offers utility. However, a near-optimal proposal offered significantly more utility than Rao-Blackwellisation in the considered scenario. The results in Section 3.4.5.3 indicate that, when the initial uncertainty is significant, the benefit of using near-optimal proposal is still apparent. However, there is an obvious benefit in adding Rao-Blackwellisation. We perceive that the reason is that near-optimal proposal provides effective positions for the samples and Rao-Blackwellisation can converge the velocity given the positions efficiently. Finally, in Section 3.4.5.4, a higher-dimensional state and observe are considered which indicates similar trends to those we have discussed above.

3.5 Conclusions

It is known that two existing techniques, the near-optimal proposal and Rao-Blackwellisation, can be used to significantly improve the particle filter. Using either technique or combining both were all mentioned in various papers. In this chapter, we specified an implementation of using both near-optimal proposal and Rao-Blackwellisation considering correlated process noise. We used a series of experiments to understand the utility of using those techniques for improving performance and maintaining efficiency. Those experiments indicate that there are scenarios in which neither technique offers significant improvements in performance. However, a near-optimal proposal is advantageous if the process noise is large and the measurements are

informative. Similarly, the unique utility of Rao-Blackwellisation is primarily associated with scenarios involving large uncertainty on the analytic sub-problem. When measurements are informative and the uncertainty of the initial velocity is pronounced, the combination of the two techniques offers improvements in performance that neither technique can achieve alone.

Chapter 4

Particle Filter for Visual Odometry

4.1 Introduction

Simultaneous localisation and mapping (SLAM) is the algorithmic process of processing sensor data to estimate the current state of a vehicle while also estimating a map. Early SLAM algorithms used the extended Kalman filter (EKF) to achieve satisfactory results [109]. The dimensionality of the stacked state comprising the vehicle state and the landmarks that parameterise the map is typically high. As a result, SLAM algorithms that use a single EKF are slow. FastSLAM [95] addressed this by exploiting the fact that, conditional on the path taken by the vehicle,[1] the estimates that relate to each landmark are conditionally independent. This use of Rao-Blackwellisation makes it possible to run a particle filter to estimate the vehicle state with each particle also running an independent Kalman filter for each landmark. Further improvements to FastSLAM focused on the particle filter part. FastSLAM 2.0 [22] extends FastSLAM by proposing particles in regions of the state-space that are probable given the received sensory data and by using an unscented Kalman filter (UKF) as an efficient and robust alternative to an EKF [110].

We now describe the application of SLAM to the visual sensory data and techniques used to enhance the accuracy of 3D mapping. State-of-the-art algorithms borrow an idea from off-line structure-from-motion approaches (which achieve an extremely low estimation error). These approaches can be categorised into feature-based (e.g., PTAM [111], ORB-SLAM [112]) and the direct method (e.g., DTAM [113], LSD-SLAM [114]). Those algorithms make use of bundle adjustments to optimise a global cost function that considers all historic camera states. Since the processing load of such bundle adjustments is large, this process is only performed relatively infrequently. This makes the difference between the SLAM and (visual) odometry estimation methods more obvious. Unlike SLAM (that aims to maximise the global mapping quality), visual odometry (VO) is a navigation method that aims to output accurate states at the time of each frame (without later correction). As explained in [115], a filter-based solution can also be appropriate when the uncertainty is pronounced and when computational resources are insufficient for bundle adjustment. Previous work has considered the fusion of data from VO and an

[1] FastSLAM also assumes that the landmarks are stationary and that the sensory data has been associated over time such that the data relating to each landmark can be identified.

Inertial Measurement Unit (IMU) (e.g., [116], [117]), though they did not consider a filter-based solution). While such a fusion is not considered in this chapter, our research will progress to consider multi-sensor fusion in the future.

In this chapter, we use FastSLAM in 3D contexts involving a monocular camera (our work is based on the description of FastSLAM in [118] and the Mono-camera framework in [23]). In such contexts, it is common to use a dynamic model for the vehicle state that involves states that describe the Cartesian position and velocity as well as other states that describe the orientation and angular velocity. If the dynamic model for the vehicle state includes velocities, the model contains a conditionally linear sub-structure. Thus, using a similar logic to that underpinning the development of FastSLAM, using Rao-Blackwellisation [92] is proposed such that a Kalman filter is used to estimate the velocity while a particle filter estimates only the vehicle position. This approach aims to decrease the dimensionality of the sub-problem that the particles need to inhabit. This idea of combining FastSLAM 2.0 and Rao-Blackwellisation has been previously proposed [119] (as part of wider research into understanding the utility of Rao-Blackwellisation [90]). However, the proposed approach considers monocular camera that does not have any control parameters gathered and the near-optimal proposal is applied which did not specified in [119].

The rest of this chapter is organised as follows. Section 4.2 reviews existing filter-based 2D SLAM solutions. The proposed RB^2-PF based 2D SLAM algorithm is then described, and a few simulations are conducted to compare it with the FastSLAM 2.0. Monocular VO is described in Section 4.3. After describing the camera models and how to extend from the proposed 2D RB^2-PF based approach to 3D, the visual features and their representation are introduced. An existence score is proposed for choosing high quality features that are considered in odometry estimation. In Section 4.3.4, several experiments are conducted that compare the proposed RB^2-PF based VO with the RB-PF based VO and the main stream Monocular SLAM algorithms on the famous public benchmarks. The conclusions are presented in Section 4.4.

4.2 Two-dimensional Simultaneous Localisation and Mapping

4.2.1 Problem Statement

We describe Ackermann model and near-constant velocity model in the following sections. Although near-constant velocity model is not usually used in conventional 2D SLAM, it is a good example for the proposed RB^2-PF algorithm and it is useful when the control parameters are unavailable.

4.2.1.1 Ackermann Steering Geometry as dynamic model

Usually, the 2D SLAM problem considers the Ackermann steering geometry as a dynamic model. The state, $s_t = [v_{x,t}, v_{y,t}, v_{\phi,t}]^T$, includes the Cartesian coordinates on x-axis, y-axis, and orientation of the vehicle, respectively. The state transition uses the control parameters: $u_t = \left[\dot{V}_t, \dot{\phi}_t\right]$. The non-linear transition model, $v_{t+1} = f(v_t, u_t)$, is described as follows:

$$\begin{bmatrix} v_{x,t+1} \\ v_{y,t+1} \\ v_{\phi,t+1}, \end{bmatrix} = \begin{bmatrix} v_{x,t} + \Delta T \cdot \dot{V}_t \cdot cos(v_{\phi,t}) \\ v_{y,t} + \Delta T \cdot \dot{V}_t \cdot sin(v_{\phi,t}) \\ v_{\phi,t} + \frac{\Delta T \cdot \dot{V}_t \cdot tan(\dot{\phi}_t)}{L} \end{bmatrix}, \tag{4.1}$$

where ΔT is the time interval between two consecutive scans, and L is the wheelbase (distance between the centres of the front and rear wheels).

The received control parameter $\hat{u}_t$ is described as follows:

$$\hat{u}_t = u_t + \omega_t, \tag{4.2}$$

where u_t is the true control parameter, and ω_t is Gaussian white noise:

$$\omega_t \sim \mathcal{N}\left(0, \begin{bmatrix} \sigma_{\dot{V}}^2 & 0 \\ 0 & \sigma_{\dot{\phi}}^2 \end{bmatrix}\right). \tag{4.3}$$

For propagating the dynamic uncertainty, the first derivative of the transition function is presented as follows:

$$\frac{\partial f(v,u)}{\partial v} = \begin{bmatrix} 1 & 0 & -\Delta T \cdot \dot{V} \cdot sin(v_\phi) \\ 0 & 1 & \Delta T \cdot \dot{V} \cdot cos(v_\phi) \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.4}$$

$$\frac{\partial f(v,u)}{\partial u} = \begin{bmatrix} \Delta T \cdot cos(v_\phi) & 0 \\ \Delta T \cdot sin(v_\phi) & 0 \\ \frac{\Delta T \cdot tan(\dot{\phi})}{L} & \frac{\Delta T \cdot \dot{V} \cdot sec^2(\dot{\phi})}{L} \end{bmatrix}. \tag{4.5}$$

4.2.1.2 Near-constant Velocity Model as dynamic model

The near-constant velocity model can also model the vehicle move in 2D SLAM. It is a linear dynamic model so that it suits the algorithm that will be proposed. When the control parameters

are unavailable, this model is effective. The state vector is defined as follows:

$$s_t = [v_{x,t}, v_{y,t}, v_{\phi,t}, \dot{v}_{x,t}, \dot{v}_{y,t}, \dot{v}_{\phi,t}]^T , \tag{4.6}$$

where $v_{x,t}, v_{y,t}, v_{\phi,t}$ are the position on the x-axis, y-axis, and the orientation of the vehicle, respectively, and $\dot{v}_{x,t}, \dot{v}_{y,t}, \dot{v}_{\phi,t}$ are the velocities of the position and orientation.

The linear state-space transition function and the definition of process noise are described in Section 3.2.1 as the case that $D = 3$.

4.2.1.3 Observations

For 2D SLAM, the sensor can detect none or multiple landmarks within the range of detection. The measurement model for each detection follows the range-bearing model (described in Section 3.2.2.2) and additionally considers the orientation of the sensor.[2] Therefore, we need to change some notations:

$$h(v_t) = \begin{bmatrix} \sqrt{(X_t^n - v_{x,t})^2 + (Y_t^n - v_{y,t})^2} \\ arctan((Y_t^n - v_{y,t}), (X_t^n - v_{x,t})) - v_{\phi,t} \end{bmatrix} + \epsilon_t, \tag{4.7}$$

where $\begin{bmatrix} X_t^n & Y_t^n \end{bmatrix}$ is the coordinate of the landmark, and ϵ_t is the measurement noise.

4.2.2 Extended Kalman Filter-SLAM

The EKF is one of the earliest solution to SLAM problems. It assumes that vehicle control noise and measurement noise are Gaussian and, most significantly, that each landmark is stationary. The state-space joins the vehicle state (Cartesian position and orientation) and landmark locations (Cartesian position) in a high-dimensional Gaussian distribution:

$$\mu_t = \begin{bmatrix} \underbrace{v_{x,t}, v_{y,t}, v_{\phi,t},}_{\text{Vehicle State:}s_t} & \underbrace{\theta_{x,t}^1, \theta_{y,t}^1}_{\text{Landmark 1 Position}} & , ..., & \underbrace{\theta_{x,t}^n, \theta_{y,t}^n}_{\text{Landmark n Position}} \end{bmatrix}^T , \tag{4.8}$$

[2] In normal 2D SLAM, the orientation of the sensor is identical to that of the vehicle. Nevertheless, we will consider another case, which will be specified in the following sections.

$$\Sigma_t = \begin{bmatrix} \sigma_{v_{x,t}v_{x,t}} & \sigma_{v_{x,t}v_{y,t}} & \sigma_{v_{x,t}v_{\phi,t}} & \sigma_{v_{x,t}\theta^1_{x,t}} & \sigma_{v_{x,t}\theta^1_{y,t}} & \cdots & \sigma_{v_{x,t}\theta^n_{x,t}} & \sigma_{v_{x,t}\theta^n_{y,t}} \\ \sigma_{v_{y,t}v_{x,t}} & \sigma_{v_{y,t}v_{y,t}v_{y,t}} & \sigma_{v_{y,t}v_{\phi,t}} & \sigma_{v_{y,t}\theta^1_{x,t}} & \sigma_{v_{y,t}\theta^1_{y,t}} & \cdots & \sigma_{v_{y,t}\theta^n_{x,t}} & \sigma_{v_{y,t}\theta^n_{y,t}} \\ \sigma_{v_{\phi,t}v_{x,t}} & \sigma_{v_{\phi,t}v_{y,t}} & \sigma_{v_{\phi,t}v_{\phi,t}} & \sigma_{v_{\phi,t}\theta^1_{x,t}} & \sigma_{v_{\phi,t}\theta^1_{y,t}} & \cdots & \sigma_{v_{\phi,t}\theta^n_{x,t}} & \sigma_{v_{\phi,t}\theta^n_{y,t}} \\ \sigma_{\theta^1_{x,t}v_{x,t}} & \sigma_{\theta^1_{x,t}v_{y,t}} & \sigma_{\theta^1_{x,t}v_{\phi,t}} & \sigma_{\theta^1_{x,t}\theta^1_{x,t}} & \sigma_{\theta^1_{x,t}\theta^1_{y,t}} & \cdots & \sigma_{\theta^1_{x,t}\theta^n_{x,t}} & \sigma_{\theta^1_{x,t}\theta^n_{y,t}} \\ \sigma_{\theta^1_{y,t}v_{x,t}} & \sigma_{\theta^1_{y,t}v_{y,t}} & \sigma_{\theta^1_{y,t}v_{\phi,t}} & \sigma_{\theta^1_{y,t}\theta^1_{x,t}} & \sigma_{\theta^1_{y,t}\theta^1_{y,t}} & \cdots & \sigma_{\theta^1_{y,t}\theta^n_{x,t}} & \sigma_{\theta^1_{y,t}\theta^n_{y,t}} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \sigma_{\theta^n_{x,t}v_{x,t}} & \sigma_{\theta^n_{x,t}v_{y,t}} & \sigma_{\theta^n_{x,t}v_{\phi,t}} & \sigma_{\theta^n_{x,t}\theta^1_{x,t}} & \sigma_{\theta^n_{x,t}\theta^1_{y,t}} & \cdots & \sigma_{\theta^n_{x,t}\theta^n_{x,t}} & \sigma_{\theta^n_{x,t}\theta^n_{y,t}} \\ \sigma_{\theta^n_{y,t}v_{x,t}} & \sigma_{\theta^n_{y,t}v_{y,t}} & \sigma_{\theta^n_{y,t}v_{\phi,t}} & \sigma_{\theta^n_{y,t}\theta^1_{x,t}} & \sigma_{\theta^n_{y,t}\theta^1_{y,t}} & \cdots & \sigma_{\theta^n_{y,t}\theta^n_{x,t}} & \sigma_{\theta^n_{y,t}\theta^n_{y,t}} \end{bmatrix}. \tag{4.9}$$

Therefore, it is apparent that when the number of landmarks increases, the state vector and the covariance matrix will increase. As mentioned, the state transition and measurement model are non-linear, and the first derivatives of those two functions (which can be found in [120]) are used for linearisation in EKF. In general, the core of EKF-SLAM is to minimise the posterior of the following distribution:

$$p\left(\Theta, s_{1:t} | z_{1:t}, u_{1:t}\right), \tag{4.10}$$

where $s_{1:t}$ denotes the vehicle (s)tate from time 1 to time t, $z_{1:t}$ denotes the measurements, $u_{1:t}$ denotes the controls, and Θ is the set of such landmark locations.

The procedure of EKF-SLAM is typical for most filter-based SLAM systems that emerge afterwards. After a control parameter is received, the vehicle propagates itself and accumulates the dynamic uncertainty. Once a sensor scan is received, the estimated measurements of landmarks in the map are predicted. Then, the sensor scans are associated with the predicted measurements. Thus, it can update the joint state (vehicle position and landmark locations) using EKF. If there are any unassociated measurements, they will be treated as new landmarks under certain conditions. The new landmarks will be added to the joint state and the covariance will be augmented. The detailed implementation and derivation are specified in [120].

The highlight of EKF-SLAM is that it keeps maintaining the entire feature map and vehicle state in the same Kalman filter update phase. The correlation between any two landmarks (i.e., the full covariance matrix) always holds explicitly, as the entire matrix that includes the invisible landmarks is updated after every sensor input. Thus, the model is deemed similar to the real world (apart from the approximations by linearisation). However, the drawback is low efficiency when the map is large, and it is less robust to the outliers.

4.2.3 FastSLAM 1.0 and 2.0

The core of FastSLAM [22,95] is the following factorisation (where the annotations indicate how part of the problem is tackled with a particle filter and part with an EKF):

$$p\left(\Theta, s_{1:t}|z_{1:t}, u_{1:t}\right) = \underbrace{p\left(s_{1:t}|z_{1:t}, u_{1:t}\right)}_{\text{Particle filter}} \prod_n \underbrace{p\left(\theta_n|s_{1:t}, z_{1:t}, u_{1:t}\right)}_{\text{Extended/unscented Kalman filter}}, \tag{4.11}$$

where most parameters are explained in Section 4.2.2. The resulting algorithm involves sampling particles, updating the Kalman filter parameters associated with each landmark, updating the particle weights, and resampling.[3] Relevant steps are briefly described.

FastSLAM 1.0 [95] considers a bootstrap particle filter such that samples are generated using:

$$q\left(s_t|s_{1:t-1}, u_{1:t}, z_{1:t}\right) = p\left(s_t|s_{t-1}, u_t\right).$$

Such a bootstrap approach is easy to implement but can lack efficiency when considering data that are informative about where the particles should be sampled. FastSLAM 2.0 [22] uses an improved proposal for sampling that capitalises on informative data. Samples are generated using $q\left(s_t|s_{1:t-1}, u_{1:t}, z_{1:t}\right) \approx p\left(s_t|s_{1:t-1}, u_{1:t}, z_{1:t}\right)$, where the approximation (necessary if the measurements are in polar coordinates) is calculated using an EKF or UKF. This approach is often referred to as using a near-optimal proposal in the broader context of particle filters, where it has found utility in several other application domains.

FastSLAM assumes that, for each particle, the uncertainty in each landmark position can be modelled as $\mathcal{N}\left(\theta_n; \hat{\theta}_{n,t}, \hat{\Sigma}_{n,t}\right)$, where $\mathcal{N}\left(x; \mu, \Sigma\right)$ denotes a Gaussian distribution with mean of μ and a covariance of Σ. Given a sampled state sequence for the vehicle, the parameters of each landmark can be updated using an EKF or UKF (assuming polar, i.e., non-linear, measurements). After that, the weight of the particles can be updated:

$$w_t = \frac{p\left(s_t|s_{t-1}, u_t\right) p\left(z_t|s_{1:t}, u_{1:t}, z_{1:t-1}\right)}{q\left(s_t|s_{1:t-1}, u_{1:t}, z_{1:t}\right)} \cdot w_{t-1}, \tag{4.12}$$

where we use the EKF or UKF to calculate a predicted measurement for the nth landmark, $\hat{z}_{n,t}$, and the associated covariance, $\hat{\Sigma}^z_{n,t}$ such that we can approximate as follows:

$$p\left(z_t|s_{1:t}, u_{1:t}, z_{1:t-1}\right) \approx \prod_n \mathcal{N}\left(z_{n,t}; \hat{z}_{n,t}, \hat{\Sigma}^z_{n,t}\right), \tag{4.13}$$

where $z_{n,t}$ is the measurement for the nth landmark.[4]

[3] In this section, the processes of data association and feature management will not be repeated since they are not related to the novelty of the proposed idea.

[4] As an aside, note that, to the best of the author's knowledge, most existing FastSLAM implementations choose to perform further approximations. We do not regard these as advantageous (in terms of computational cost or estimation accuracy). All the terms in (4.12) will be readily accessible in a SLAM implementation. The approximations are:

$$w_t \approx p\left(z_t|s_{1:t-1}, u_{1:t}, z_{1:t-1}\right) \cdot w_{t-1} \tag{4.14}$$

$$\approx \prod_n N\left(z_{n,t}; \tilde{z}_{n,t}, \tilde{\Sigma}^z_{n,t}\right) \cdot w_{t-1} \tag{4.15}$$

One can improve on the original FastSLAM 2.0 by replacing the use of derivatives in the EKF with the use of an unscented transformation (an approach that has been shown to improve estimation consistency [82] in other contexts). Unscented FastSLAM (UFastSLAM) [110, 121] adopts this approach and specifically uses the scaled unscented transformation (SUT) [122], which we use to generate the results in Section 4.2.5.

The final stage of the particle filter is resampling. This is used to alleviate the degeneracy phenomenon: without resampling, after a few iterations, a small number of particles have much larger weights than other particles with most particles contributing almost nothing to the particle filter's estimation (which are a weighted sum across particles). Resampling probabilistically discards the particles with low weights and replicates particles with high weights. The details are described elsewhere (e.g., in [80]).

4.2.4 The RB^2-PF based 2D Simultaneous Localisation and Mapping

We describe how to sample the particles and perform the weight update. Reminding that all the linearisation parts use the unscented transformation rather than the first derivative in FastSLAM 1.0 and 2.0. The other steps (e.g., data association and resampling) are unchanged relative to existing FastSLAM 2.0 implementations and are described elsewhere [123, 124].

4.2.4.1 Sampling Vehicle Position with Near-optimal Proposal

The calculation of a near-optimal proposal is similar to FastSLAM 2.0 except for two things. The particles only sample the vehicle position, and the uncertainty in velocity that is estimated by the Kalman filter needs to be considered. The process follows the proposed standard particle filter using Rao-Blackwellisation and the near-optimal proposal in Chapter 3 and is described as follows.

We first calculate the predicted position $s^p_{t|t-1}$ and associated covariance $\Sigma^p_{t|t-1}$ using (3.64). For sampling the vehicle position using the near-optimal proposal, it is necessary to obtain the associated observations. In 2D SLAM, it can be done using several approaches, such as nearest neighbour (NN) that is described in Section B.1, probabilistic data association Filter (PDAF) that is described in Section B.2, joint compatibility branch and bound (JCBB) [125]. We choose to use the nearest neighbour (NN) method in the following experiments. The near-optimal proposal is calculated by Algorithm 5 with the values of $s^p_{t|t-1}$, $\Sigma^p_{t|t-1}$ and the associated measurements.

where $\tilde{z}_{n,t}$ and $\tilde{\Sigma}^z_{n,t}$ involve using the EKF or UKF to predict the measurement and associated covariance for the nth landmark given only s_{t-1} (and not also the sampled value of s_t).

4.2.4.2 Update Vehicle Velocity

The velocity update with the Kalman filter is exactly the same as the generic particle filter using Rao-Blackwellisation and near-optimal proposal described in Section 3.4.3.2.

4.2.4.3 Update Particle Weights

The weight update is derived using a similar logic to that used in FastSLAM 2.0, albeit with u_t dropped from the notation (or it is also equivalent to Section 3.4.3.1):

$$w_t = w_{t-1} \cdot \frac{p\left(s_t^p | s_{1:t-1}^p, z_{1:t-1}\right) p\left(z_t | s_{1:t}^p, z_{1:t-1}\right)}{q\left(s_t^p | s_{1:t-1}^p, z_{1:t}\right)}. \tag{4.16}$$

4.2.4.4 Algorithm Summary

The framework of the proposed algorithm is similar to FastSLAM 2.0, being its improvement, except for adopting the near-constant velocity model and a separate step for updating the velocity. The pseudo-code of the proposed algorithm is shown in Algorithm 9.

Algorithm 9 The pseudo-code of RB^2-PF based 2D SLAM.

Initialise the particles (including the mean and covariance of position, the mean and covariance of velocity, and the particle weights).
while Sensor scan receives **do**
  Vehicle position prediction (Section 4.2.4.1).
  Landmark observation prediction (Section 4.2.1.3).
  Landmark association (with nearest neighbour Section B.1).
  Sample the vehicle position (Section 4.2.4.1).
  Perform vehicle velocity update (Section 4.2.4.2).
  Perform landmark location update (refer to [110]).
  Update particle weights (Section 4.2.4.3).
  for All new landmark **do**
    Initialise the new landmark (refer to [110]).
  end for
  if Effective sample size is low **then**
    Perform resampling (refer to [80]).
  end if
end while

4.2.5 Experiments with RB^2-PF for two dimensional SLAM

4.2.5.1 Simulation Description

The focus is on comparing the performance of the proposed algorithm (RB^2-PF) with UFastSLAM [121] (which is the unscented RB-PF based approach), which we perceive to be an appropriate baseline FastSLAM implementation. To reflect solely the performance of two algorithms, we first assume that the data associations are always true. Then, we consider a more realistic situation in which the data associations are unknown, and mis-detections can happen with a fixed probability. The experiments consist of two scenarios. One hundred Monte-Carlo simulations are used to provide fair conclusions in the random assumptions. Both simulations share the basic concept that a moving vehicle with a range-bearing sensor can detect several stationary landmarks in a sequence of time steps. The dynamic of the vehicle varies in two scenarios. We describe the two scenarios as follows in turn.

Scenario 1 A classic SLAM simulation scenario is partially adopted and can be briefly described as follows.

- The vehicle motion follows the Ackermann model.
- The sensor is fixed on the vehicle, such that the orientation of the sensor is identical to the orientation of the vehicle.
- The trajectory of the vehicle is from the initial point to a number of points in sequence, such that the trajectory is fixed during the Monte-Carlo simulations.
- The locations of the landmarks are irregularly pre-defined along the trajectory of the vehicle, and they will not change during the Monte-Carlo simulations.
- The measurement noise is the only random parameter in the Monte-Carlo simulations.

The trajectory and the locations of landmarks are shown in Figure 4.1. To fit this scenario to the proposed SLAM algorithm, a few changes (mostly making the problem more challenging) have been made as follows:

- Although the Ackermann model is used for the dynamic model, the parameters of velocity and steering control are all unknown (such that a near-constant velocity model is applied to the position and velocity of the vehicle).
- Although the sensor orientation is related to that of the vehicle, we do not assume this as a prior knowledge (such that we still assume a near-constant velocity model for the orientation).
- We ensure that the sensor can detect many landmarks at every time step.

It is aware that the near-constant velocity model is not usually considered in traditional SLAM cases. However, this simulation is to investigate whether RB^2-PF can outperform unscented RB-PF in the problem of SLAM. Therefore, it is sensible to conduct a well-known SLAM simulation with some naive but essential modifications.

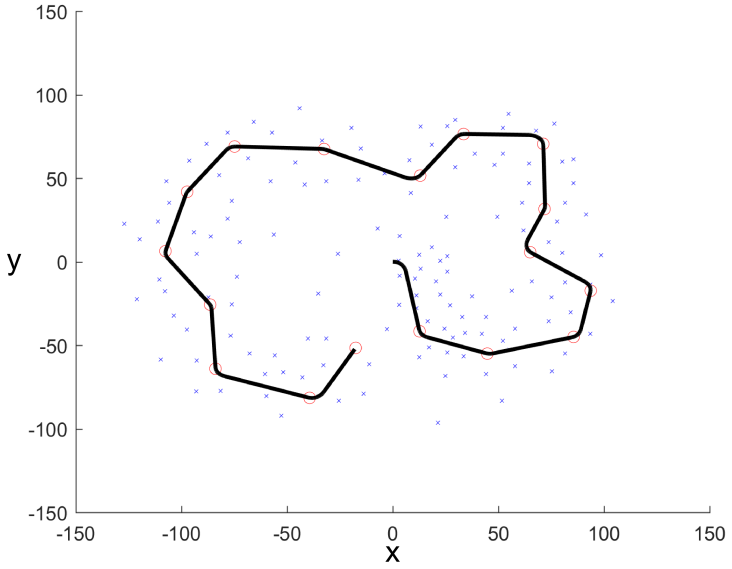


FIGURE 4.1: The vehicle trajectory and landmark locations in simulation Scenario 1.

.

Scenario 2 The vehicle movements are free from the constraints, such as steering and orientation. It is more sensible for testing the proposed approach. The brief descriptions are as follows:

- The vehicle dynamic s follows the near-constant velocity model (such that the trajectories are not fixed for each of Monte-Carlo simulations).
- The sensor orientation follows the near-constant velocity model as well.
- The landmarks are formed as a grid within the range that the vehicles are possibly staying in. Every landmark is still stationary during the Monte-Carlo simulations.
- The measurement noise is still Gaussian.

Some examples of the trajectories and the locations of landmarks are shown in Figure 4.2.

4.2.5.2 Experiment Results Using Scenario 1: Fixed Vehicle Trajectory (known data association)

Parameters for the simulation and the RB^2-PF filter

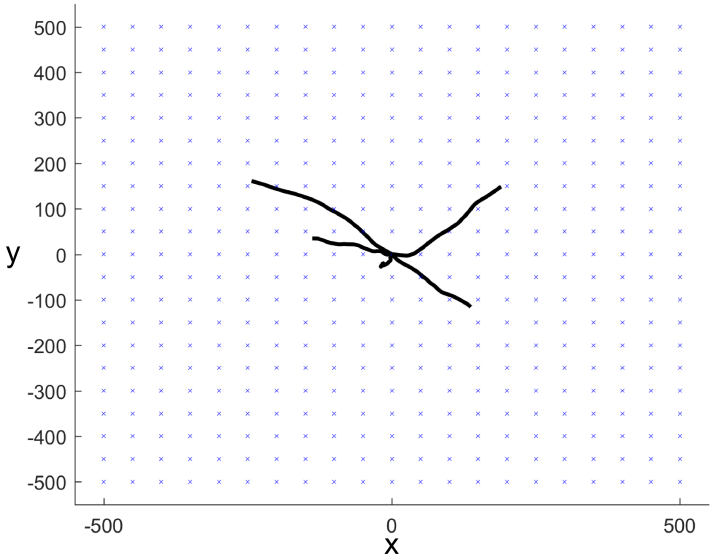


FIGURE 4.2: A few examples of random vehicle trajectories and landmark locations in simulation Scenario 2.

.

For both simulation and the filter, the sampling rate of the range-bearing sensor is 10 Hz. The measurement noise is described in (3.13), and we assume $R = diag(\left[0.25, \frac{1\times\pi}{180}\right]^2)$.

As the true trajectory of the vehicle stays the same during the Monte-Carlo simulations, the following parameters are only for the filter. The process noise is described in (3.8), and we assume $\sigma = diag([0.1, 0.1, 0.01, 0.1, 0.1, 0.01]^2)$. We assume that the vehicle can observe the landmarks up to a maximum range of 30 m and within a 180° field of view centred on the current heading.

Results and Discussion

The experiment is based on Scenario 1, which is very similar to a real case in which a sensor is fixed on a vehicle/robot. In such a case, although the dynamic model of the filter does not exactly match the simulation problem, the experiment can somehow show the different performance between RB-PF and RB^2-PF in the fashion that robotic researchers often use.

Table 4.1 shows the average and median root mean square error (RMSE) after 100 simulations and the numbers of failed estimations are also presented. The failed estimation is the case in which all the particles have zero weight.[5] From these figures, we can see the proposed RB^2-PF had many more valid results when the number of particles is low. By focusing on the average and median RMSEs, the RB^2-PF based SLAM algorithm always outperformed the RB-PF based algorithm. When there are five particles, the improvement was nearly 50%. Obviously,

[5]Although it can be fixed using the logarithm of the probabilities, it still reflects that the filter is weak on addressing the problem.

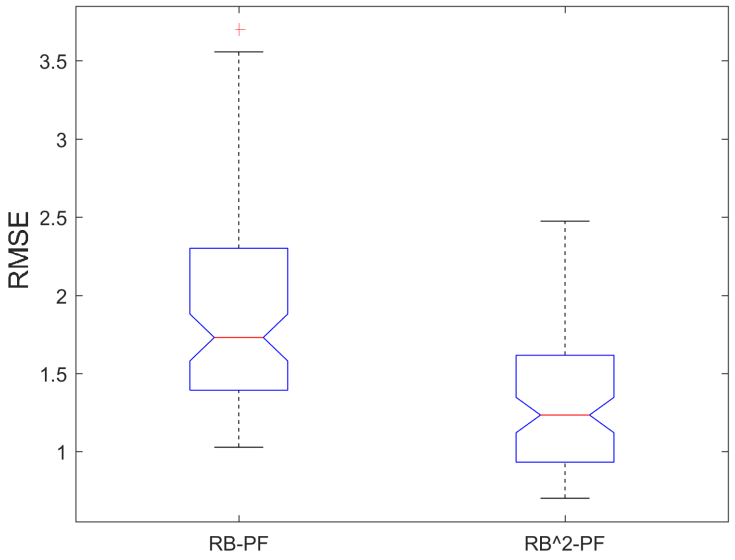
	RB-PF			RB^2-PF		
Number of Particles	Ave. RMSE	Med. RMSE	(*)	Ave. RMSE	Med. RMSE	(*)
5	4.61	4.15	90	2.43	2.18	3
10	3.02	2.63	42	2.17	1.98	0
25	2.26	2.07	1	1.63	1.60	0
50	1.97	1.73	0	1.37	1.23	0
100	1.60	1.43	0	1.21	1.05	0

Column (*) shows the number of failed estimations.

TABLE 4.1: The average (Ave.) and the median (Med.) root mean square error (RMSE) [meter] of position from 100 Monte-Carlo simulations using different numbers of particles. The scenario described in Section 4.2.5.2 was used for the simulations.

the improvement percentage decreases when more particles are used in the filter. When using 100 particles, the rate decreases to nearly 25%. That is because, when the effective sample size of RB-PF goes up, the problem can be well addressed. When enough (which could be a very large number) particles are used, the performances of RB-PF and RB^2-PF can be very similar.

We also focus on the distribution of the errors from 100 simulations. Figure 4.3 shows a case in which 50 particles are used. Note that we have ignored 5% of the largest and 5% of the smallest errors. The variance of the errors from RB-PF is apparently larger than that of the errors from RB^2-PF. Comparing the upper bounds, it is clear that RB^2-PF yields more reliable estimations.



The results are from 100 Monte-carlo runs. 50 particles are used for the particle filter.

FIGURE 4.3: The distribution of root mean square errors (RMSEs) of the estimated vehicle positions. The simulation considers the scenario described in Section 4.2.5.2.

	RB-PF			RB^2-PF		
Number of Particles	Ave. RMSE	Med. RMSE	(*)	Ave. RMSE	Med. RMSE	(*)
5	7.79	7.39	12	5.88	5.43	8
10	5.40	4.94	7	3.93	3.53	5
25	3.03	2.92	3	2.44	2.35	1
50	2.09	1.86	0	1.57	1.44	0
100	1.35	1.23	0	1.08	0.94	0

Column (*) shows the number of failed estimations.

TABLE 4.2: The average (Ave.) and median (Med.) root mean square error (RMSE) from 100 Monte-Carlo simulations with different numbers of particles. The scenario described in Section 4.2.5.3 was used for the simulations.

4.2.5.3 Experiment Results Using Scenario 2: Random Vehicle Trajectory (known data association)

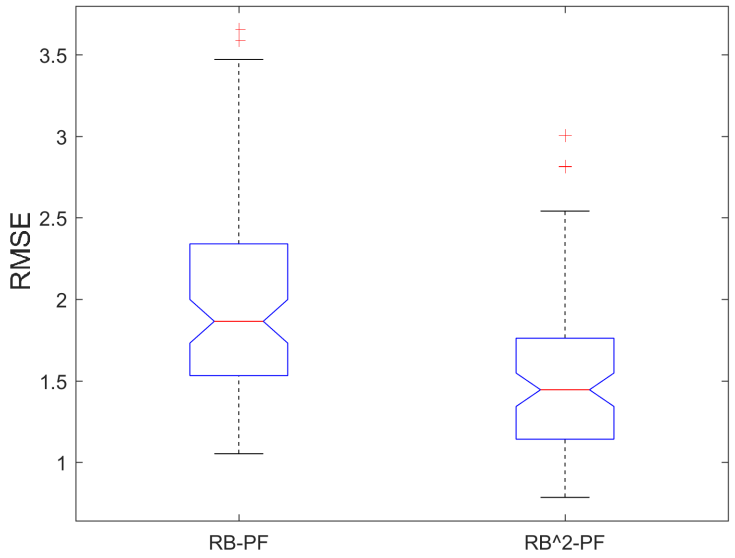
Parameters for the simulations and RB^2-PF filter

For both the simulations and filter, we use identical parameters. The sampling rate of the range-bearing sensor is 10 Hz and there are 300 iterations in each simulation. The initial state of the vehicle is $s_0 = [0, 0, 0, 0.5, 0.5, 0]^T$. The process noise is $\sigma = diag([0.1, 0.1, 0.01, 0.1, 0.1, 0.01]^2)$. We assume that the vehicle can only observe landmarks up to a maximum range of 150 m and within a 180° field of view centred on the current heading. The measurement noise is $R = diag([2, \frac{1\times\pi}{180}]^2)$.

Results and Discussion

The experiment follows Scenario 2. The performances of RB^2-PF and RB-PF on the simulations are shown in Table 4.2. We notice that, no matter how many particles are used, the RMSEs from RB^2-PF are always smaller than those for RB-PF. Figure 4.4 shows the distribution of RMSEs when 50 particles are used. After eliminating the highest and lowest 5 percentile of errors, RB-PF still creates more high-error outliers than RB^2-PF. The boxplot also shows that, with the random process noise and measurement noise, the proposed algorithm yields smaller variance of errors across the simulations. All the above indicates the improvements in both performance and reliability via using RB^2-PF. Similar to the discovery in Section 4.2.5.2, while the number of particles increases, the improvement rate becomes smaller. It is worth discussing which algorithm should be chosen: RB-PF with more particles or RB^2-PF with less particles. Although the cost of memory of the difference is trivial in modern computers, the computational cost will be our main concern.

The reports on the runtime consumed by the two algorithms are shown in Table 4.3. Perhaps unsurprisingly, RB^2-PF runs more slowly than RB-PF; however, the gap is small. This is because, although RB^2-PF involves an extra Kalman filter for each particle (relative to RB-PF), the dimensionality that the particles need to inhabit is halved. Recalling that we used unscented



The results are from 100 Monte-carlo runs. 50 particles are used for the particle filter.

FIGURE 4.4: The distribution of root mean square errors (RMSEs) of the estimated vehicle positions. The simulation considers the scenario described in Section 4.2.5.3.

transformation for linearisation with every Kalman filter, the result is that fewer sigma points are needed by the UKF associated with the near-optimal proposal. It is likely that, when an EKF is used to calculate a near-optimal proposal, the time difference would be enlarged. However, previous work [110,121] has highlighted the improvement in accuracy offered by the UKF; hence, the UKF is used here.[6]

The above discussion suggests the general idea that the RB^2-PF based 2D SLAM approach is obviously recommended, as the RB^2-PF outperforms the other approach along with the increasing number of particles and consumes similar time for such problems. If we otherwise only care about the extreme performance and do not mind using an overwhelming number of particles, RB^2-PF cannot perform better compared to RB-PF.

4.2.5.4 Experiment Results Using Scenario 2: Random Vehicle Trajectory (unknown data association and mis-detection)

Parameters for the simulations and RB^2-PF filter

For both the simulation and filter, we use identical parameters, and they equal those used in Section 4.2.5.3. Moreover, the data association is unknown, and we use the nearest-neighbour

[6] In fact, there are also potential theoretical concerns with using an EKF to define a proposal in a particle filter. An EKF typically underestimates the covariance such that a proposal defined using an EKF can be insufficiently heavy tailed to be theoretically valid. The potential empirical effect is that using more samples with an EKF-based proposal is not guaranteed to improve estimation accuracy.

Number of Particles	RB-PF [s]	RB^2-PF [s]
5	6.62	6.81
10	12.98	13.35
25	32.10	33.04
50	63.96	65.83
100	127.75	131.64

TABLE 4.3: The average time taken for RB-PF and RB^2-PF based SLAM to perform 100 simulation runs with different numbers of particles. The scenario described in Section 4.2.5.3 was used for the simulations.

	RB-PF			RB^2-PF		
Number of Particles	Ave. RMSE	Med. RMSE	(*)	Ave. RMSE	Med. RMSE	(*)
5	9.88	6.90	0	10.09	7.11	0
10	6.01	4.01	0	4.96	3.31	0
25	2.69	2.15	0	2.30	2.00	0
50	1.86	1.58	0	1.61	1.43	0
100	1.48	1.12	0	1.19	0.98	0

Column (*) shows the number of failed estimations.

TABLE 4.4: The average (Ave.) and median (Med.) root mean square error (RMSE) [meter] of position from 100 Monte-Carlo simulations using different numbers of particles. The scenario described in Section 4.2.5.4 was used for the simulations.

method to associate the landmarks with the estimated map. The mis-detection can happen with a probability of 10%.

Results and Discussion

The mean and median estimations RMSE over 100 Monte-Carlo simulations are shown in Table 4.4. Mostly, we have very similar conclusions with the scenario that data association is always known. The experiment shows that the improvement still holds when the problem is complicated and closer to reality. Compared to Table 4.2, we notice an interesting situation, which is that none of the simulations failed. The reason is that, if the estimated vehicle states from the particles all go wrong, the predicted measurements of the landmark should be all wrong as well. As the data association is unknown, the predicted measurements will not be forced to associate with the observed landmarks. Those observed landmarks are likely to be treated as newly detected ones. Thus, the measurement probabilities with those particles will not be zero; hence, the weights are non-zero.

4.3 Monocular Visual Odometry

4.3.1 Problem Statement

4.3.1.1 Camera State Model

We use the definition of the camera state, which is widely used in Monocular SLAM/VO (e.g., [23, 118]) as follows.

$$s_t = \begin{bmatrix} v_t & o_t & \dot{v}_t & \dot{o}_t \end{bmatrix}^T, \tag{4.17}$$

where v_t is the Cartesian position of the camera, o_t is the orientation of the camera, which is represented using (q)uaternions, $\dot{v}_t$ is the Cartesian velocity, and $\dot{o}_t$ is the orientation velocity, which is represented using (E)uler angles. In order to illustrate the property of orientation and its velocity, we use o_t^q and $\dot{o}_t^E$ instead of o_t and $\dot{o}_t$, such that:

$$s_t = \begin{bmatrix} v_t & o_t^q & \dot{v}_t & \dot{o}_t^E \end{bmatrix}^T. \tag{4.18}$$

The dimensions of v_t, o_t^q, $\dot{v}_t$, $\dot{o}_t^E$ are 3, 4, 3, and 3, respectively. Therefore, s_t is a 13-dimensional parameter.

4.3.1.2 Camera Dynamic Model

We consider the camera dynamic model as follows:

$$\begin{bmatrix} v_{t+1} \\ o_{t+1}^q \\ \dot{v}_{t+1} \\ \dot{o}_{t+1}^E \end{bmatrix} = \begin{bmatrix} v_t + \dot{v}_{t+1} \cdot \Delta T \\ A(o_t^q, G(\dot{o}_t^E \cdot \Delta T)) \\ \dot{v}_t \\ \dot{o}_t^E \end{bmatrix} + \omega_t, \tag{4.19}$$

where ΔT is the time interval between two frames, and ω_t is the Gaussian white noise as the process noise.

The quaternion multiplication function $A(.)$ is defined as:

$$A\left(\begin{bmatrix}a_1\\b_1\\c_1\\d_1\end{bmatrix},\begin{bmatrix}a_2\\b_2\\c_2\\d_2\end{bmatrix}\right)=\begin{bmatrix}a_1a_2-b_1b_2-c_1c_2-d_1d_2\\a_1b_2+b_1a_2+c_1d_2-d_1c_2\\a_1c_2-b_1d_2+c_1a_2+d_1b_2\\a_1d_2+b_1c_2-c_1b_2+d_1a_2\end{bmatrix}, \tag{4.20}$$

and its first-order partial derivative/Jacobian matrix (which will be used for linearisation later) can be derived as follows:

$$\frac{\partial A\left(\left[a_1,b_1,c_1,d_1\right]^T,\left[a_2,b_2,c_2,d_2\right]^T\right)}{\partial\left[a_2,b_2,c_2,d_2\right]^T}=\begin{bmatrix}a_1 & -b_1 & -c_1 & -d_1\\ b_1 & a_1 & -d_1 & c_1\\ c_1 & d_1 & a_1 & -b_1\\ d_1 & -c_1 & b_1 & a_1\end{bmatrix}. \tag{4.21}$$

Moreover, $G(.)$ is the transformation function from the Euler angle to quaternion, which is defined as (axis rotation sequence $x-y-z$ is used; for more information, refer to [126]):

$$o^q=G(\begin{bmatrix}o_1^E\\o_2^E\\o_3^E\end{bmatrix})=\begin{bmatrix}-sin(\frac{o_1^E}{2})sin(\frac{o_2^E}{2})sin(\frac{o_3^E}{2})+cos(\frac{o_1^E}{2})cos(\frac{o_2^E}{2})cos(\frac{o_3^E}{2})\\ sin(\frac{o_1^E}{2})cos(\frac{o_2^E}{2})cos(\frac{o_3^E}{2})+sin(\frac{o_2^E}{2})sin(\frac{o_3^E}{2})cos(\frac{o_1^E}{2})\\ -sin(\frac{o_1^E}{2})sin(\frac{o_3^E}{2})cos(\frac{o_2^E}{2})+sin(\frac{o_2^E}{2})cos(\frac{o_1^E}{2})cos(\frac{o_3^E}{2})\\ sin(\frac{o_1^E}{2})sin(\frac{o_2^E}{2})cos(\frac{o_3^E}{2})+sin(\frac{o_3^E}{2})cos(\frac{o_1^E}{2})cos(\frac{o_2^E}{2})\end{bmatrix}, \tag{4.22}$$

and its first-order partial derivative is as follows:

$$\frac{\partial o^q}{\partial o^E}=\frac{\partial\left[o_1^q,o_2^q,o_3^q,o_4^q\right]}{\partial\left[o_1^E,o_2^E,o_3^E\right]}=\begin{bmatrix}\frac{\partial o_1^q}{\partial o_1^E} & \frac{\partial o_1^q}{\partial o_2^E} & \frac{\partial o_1^q}{\partial o_3^E}\\ \frac{\partial o_2^q}{\partial o_1^E} & \frac{\partial o_2^q}{\partial o_2^E} & \frac{\partial o_2^q}{\partial o_3^E}\\ \frac{\partial o_3^q}{\partial o_1^E} & \frac{\partial o_3^q}{\partial o_2^E} & \frac{\partial o_3^q}{\partial o_3^E}\\ \frac{\partial o_4^q}{\partial o_1^E} & \frac{\partial o_4^q}{\partial o_2^E} & \frac{\partial o_4^q}{\partial o_3^E}\end{bmatrix}, \tag{4.23}$$

where

$$\frac{\partial o_1^q}{\partial o_1^E} = -\frac{1}{2}cos(\frac{o_1^E}{2}) \cdot sin(\frac{o_2^E}{2}) \cdot sin(\frac{o_3^E}{2}) - \frac{1}{2}sin(\frac{o_1^E}{2}) \cdot cos(\frac{o_2^E}{2}) \cdot cos(\frac{o_3^E}{2}), \tag{4.24}$$

$$\frac{\partial o_1^q}{\partial o_2^E} = -\frac{1}{2}sin(\frac{o_1^E}{2}) \cdot cos(\frac{o_2^E}{2}) \cdot sin(\frac{o_3^E}{2}) - \frac{1}{2}cos(\frac{o_1^E}{2}) \cdot sin(\frac{o_2^E}{2}) \cdot cos(\frac{o_3^E}{2}), \tag{4.25}$$

$$\frac{\partial o_1^q}{\partial o_3^E} = -\frac{1}{2}sin(\frac{o_1^E}{2}) \cdot sin(\frac{o_2^E}{2}) \cdot cos(\frac{o_3^E}{2}) - \frac{1}{2}cos(\frac{o_1^E}{2}) \cdot cos(\frac{o_2^E}{2}) \cdot sin(\frac{o_3^E}{2}), \tag{4.26}$$

$$\frac{\partial o_2^q}{\partial o_1^E} = +\frac{1}{2}cos(\frac{o_1^E}{2}) \cdot cos(\frac{o_2^E}{2}) \cdot cos(\frac{o_3^E}{2}) - \frac{1}{2}sin(\frac{o_2^E}{2}) \cdot sin(\frac{o_3^E}{2}) \cdot sin(\frac{o_1^E}{2}), \tag{4.27}$$

$$\frac{\partial o_2^q}{\partial o_2^E} = -\frac{1}{2}sin(\frac{o_1^E}{2}) \cdot sin(\frac{o_2^E}{2}) \cdot cos(\frac{o_3^E}{2}) + \frac{1}{2}cos(\frac{o_2^E}{2}) \cdot sin(\frac{o_3^E}{2}) \cdot cos(\frac{o_1^E}{2}), \tag{4.28}$$

$$\frac{\partial o_2^q}{\partial o_3^E} = -\frac{1}{2}sin(\frac{o_1^E}{2}) \cdot cos(\frac{o_2^E}{2}) \cdot sin(\frac{o_3^E}{2}) + \frac{1}{2}sin(\frac{o_2^E}{2}) \cdot cos(\frac{o_3^E}{2}) \cdot cos(\frac{o_1^E}{2}), \tag{4.29}$$

$$\frac{\partial o_3^q}{\partial o_1^E} = -\frac{1}{2}cos(\frac{o_1^E}{2}) \cdot sin(\frac{o_3^E}{2}) \cdot cos(\frac{o_2^E}{2}) - \frac{1}{2}sin(\frac{o_2^E}{2}) \cdot sin(\frac{o_1^E}{2}) \cdot cos(\frac{o_3^E}{2}), \tag{4.30}$$

$$\frac{\partial o_3^q}{\partial o_2^E} = +\frac{1}{2}sin(\frac{o_1^E}{2}) \cdot sin(\frac{o_3^E}{2}) \cdot sin(\frac{o_2^E}{2}) + \frac{1}{2}cos(\frac{o_2^E}{2}) \cdot cos(\frac{o_1^E}{2}) \cdot cos(\frac{o_3^E}{2}), \tag{4.31}$$

$$\frac{\partial o_3^q}{\partial o_3^E} = -\frac{1}{2}sin(\frac{o_1^E}{2}) \cdot cos(\frac{o_3^E}{2}) \cdot cos(\frac{o_2^E}{2}) - \frac{1}{2}sin(\frac{o_2^E}{2}) \cdot cos(\frac{o_1^E}{2}) \cdot sin(\frac{o_3^E}{2}), \tag{4.32}$$

$$\frac{\partial o_4^q}{\partial o_1^E} = +\frac{1}{2}cos(\frac{o_1^E}{2}) \cdot sin(\frac{o_2^E}{2}) \cdot cos(\frac{o_3^E}{2}) - \frac{1}{2}sin(\frac{o_3^E}{2}) \cdot sin(\frac{o_1^E}{2}) \cdot cos(\frac{o_2^E}{2}), \tag{4.33}$$

$$\frac{\partial o_4^q}{\partial o_2^E} = +\frac{1}{2}sin(\frac{o_1^E}{2}) \cdot cos(\frac{o_2^E}{2}) \cdot cos(\frac{o_3^E}{2}) - \frac{1}{2}sin(\frac{o_3^E}{2}) \cdot cos(\frac{o_1^E}{2}) \cdot sin(\frac{o_2^E}{2}), \tag{4.34}$$

$$\frac{\partial o_4^q}{\partial o_3^E} = -\frac{1}{2}sin(\frac{o_1^E}{2}) \cdot sin(\frac{o_2^E}{2}) \cdot sin(\frac{o_3^E}{2}) + \frac{1}{2}cos(\frac{o_3^E}{2}) \cdot cos(\frac{o_1^E}{2}) \cdot cos(\frac{o_2^E}{2}). \tag{4.35}$$

4.3.1.3 Feature Position with Pin-hole Camera Model

The monocular pin-hole camera model without both radial and tangential distortion (e.g., [24]) is considered. The camera model is defined as follows:

$$\begin{bmatrix} h_x^{(i)} \\ h_y^{(i)} \\ h_z^{(i)} \end{bmatrix} = (R(o_t))^T \cdot \left(\begin{bmatrix} X^{(i)} \\ Y^{(i)} \\ Z^{(i)} \end{bmatrix} - v_t \right), \tag{4.36}$$

where $\begin{bmatrix} h_x^{(i)}, & h_y^{(i)}, & h_z^{(i)} \end{bmatrix}^T$ is the position of feature point (i) in a 3D Cartesian space with respect to the current camera state, $R(.)$ is a transformation function that transforms quaternions to a rotation matrix, and $\begin{bmatrix} X^{(i)} & Y^{(i)} & Z^{(i)} \end{bmatrix}$ is the position of landmark (i) in the global Cartesian space.

For the point on the image, we have:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} u_0 - f_x \cdot \frac{h_x^{(i)}}{h_z^i} \\ v_0 - f_y \cdot \frac{h_y^{(i)}}{h_z^i} \end{bmatrix}, \tag{4.37}$$

where $\begin{bmatrix} u_i, & v_i \end{bmatrix}^T$ is the coordinator of the feature point on the image, $\begin{bmatrix} u_0, & v_0 \end{bmatrix}^T$ is the principle point of the camera, and $\begin{bmatrix} f_x, & f_y \end{bmatrix}^T$ is focal length of the camera.

It is known that a camera usually introduces distortion. Pronounced distortion can damage the consistency of the feature descriptor, as it is always assumed that a landmark feature is invariant over time. We only use the rectified images from public benchmarks, and such discussion will not be expanded. For more information about calibration, [127] is a recommended literature.

4.3.2 The RB^2-PF for Monocular Visual Odometry

The derivation of RB^2-PF is almost the same using 2D SLAM (as in Section 4.2.4) except that the dimension of the problem is increased and the dynamic of the orientation of the camera is non-linear. The modification will be described in Section 4.3.2.1. We highlight that the the non-linear dynamic of camera orientation is handled by an EKF. Section 4.3.2.2 will describe how the modification can be used in RB^2-PF for VO.

4.3.2.1 Considering the Non-linear Orientation Dynamics

In the proposed system, the camera orientation still follows the near-constant velocity model, which means most of the derivations for 2D SLAM (under the assumption of near-constant velocity model) can still be used. However, the transformation between the quaternion and Euler angle (including the representation of orientation and velocity) is non-linear, and the time propagation of the orientation that is represented using the quaternion is also non-linear. As the Rao-Blackwellisation derivation (in Chapter 3) is under the assumption of a linear dynamic model, we choose to use the first derivative to linearise all the above transformations. Thus, the transition matrix (3.7) and the process noise matrix (3.8) must be modified. Reminding that the quaternions are known to be four-dimensional and the Euler angles are known to be three-dimensional.

The transition matrix of the camera orientation, as expressed using quaternions can be written as follows:

$$o_{t+1}^q = A(o_t^q, \Delta o_t^q), \tag{4.38}$$

where o_t^q is the previous orientation, Δo_t^q is the orientation increment, and $A(.)$ is the quaternion multiplication 4.20.

Considering the angular velocity as described using Euler angles, we have:

$$\Delta o_t^q = G(\Delta o_t^E), \tag{4.39}$$

where $G(.)$ is the transformation function from Euler angles to quaternions (details can be found in [126]). Note that the increment $\Delta o_t^E = \dot{o}_t^E \cdot \Delta T$.

The transition matrix (which is a 7×7 matrix) for the orientation is approximated as follows:

$$F^O \approx \begin{bmatrix} \mathbb{I}_{4\times4} & \xi_t \\ \mathbb{O}_{3\times4} & \mathbb{I}_{3\times3} \end{bmatrix}, \tag{4.40}$$

where ξ_t (which is a 4×3 matrix) is calculated as follows:

$$\xi_t = \frac{\partial A(o_t^q, \Delta o_t^q)}{\partial \Delta o_t^q} \cdot \frac{\partial G(\dot{o}_t^E \cdot \Delta T)}{\partial \dot{o}_t^E}. \tag{4.41}$$

Note that the transition matrix is an approximation of the non-linear time update of orientation. The approximation is likely to be small with the small ΔT relevant to monocular VO ($\Delta T = 1/30s$ in our experiments).

The approximated process noise of orientation can be calculated by (4.42), which uses the first derivative for linearisation.

$$Q_t^O = \tau_t \cdot Q_t^E \cdot \tau_t^T, \tag{4.42}$$

where Q_t^E is the process noise using Euler angles, which has the same form as in (3.8) and:

$$\tau = \begin{bmatrix} \hat{\xi}_t & \mathbb{O}_{4\times3} \\ \mathbb{O}_{3\times3} & \mathbb{I}_{3\times3} \end{bmatrix}, \tag{4.43}$$

where $\hat{\xi}_t$ is from (4.41) with $\Delta T = 1$ and τ is a 7×6 matrix.

4.3.2.2 Algorithm Summary

For VO, the state of camera[7] integrates position, orientation, and their velocities:

$$s_t = \left[\begin{array}{cc:cc} v_t & o_t^q & \dot{v}_t & \dot{o}_t^E \end{array} \right]^T. \tag{4.44}$$

[7] We use s_t for the system state, so it is consistent with Chapter 3.

We put the above camera state and measurement model (Section 4.3.1.3) into the RB^2-PF framework described in Section 3.4.4 or Section 4.2.4. Thus, several core steps of visual odometry is that duplicate the whole processes of RB^2-PF. We only substitute v_t with s_t and use the pin-hole camera model as mentioned. Furthermore, the transition matrix and process noise matrix with the linearisation are modified as follows:

$$F = \left[\begin{array}{cc:cc} F_{11}^P & \mathbb{O}_{3\times4} & F_{12}^P & \mathbb{O}_{3\times3} \\ \mathbb{O}_{4\times3} & F_{11}^O & \mathbb{O}_{4\times3} & F_{12}^O \\ \hdashline F_{21}^P & \mathbb{O}_{3\times4} & F_{22}^P & \mathbb{O}_{3\times3} \\ \mathbb{O}_{3\times3} & F_{21}^O & \mathbb{O}_{3\times3} & F_{22}^O \end{array}\right], \tag{4.45}$$

$$Q = \left[\begin{array}{cc:cc} Q_{11}^P & \mathbb{O}_{3\times4} & Q_{12}^P & \mathbb{O}_{3\times3} \\ \mathbb{O}_{4\times3} & Q_{11}^O & \mathbb{O}_{4\times3} & Q_{12}^O \\ \hdashline Q_{21}^P & \mathbb{O}_{3\times4} & Q_{22}^P & \mathbb{O}_{3\times3} \\ \mathbb{O}_{3\times3} & Q_{21}^O & \mathbb{O}_{3\times3} & Q_{22}^O \end{array}\right], \tag{4.46}$$

where F^P is the position transition matrix (3.7), Q^P is the position process noise (3.8), and F^O and Q^O are from (4.40) and (4.42).

A brief implementation of one iteration of the proposed algorithm is shown in Algorithm 10. Some of the steps (e.g., resampling) are identical to those in the classical FastSLAM 2.0. The remaining details are described in the following section.

4.3.3 Visual Landmarks and Features

Each landmark contains the position and feature information. This section will introduce how the landmark positions are represented. The initialisation and matching of the features (of the landmarks) will be described and existence score will be proposed to maintain the landmarks.

4.3.3.1 Landmark Position Representation

For the Monocular SLAM/VO problem, the depth of a landmark cannot be obtained. If we keep using Cartesian coordinates to represent a landmark like that in 2D SLAM, the uncertainty of the landmark position needs to be transferred between polar coordinates and Cartesian coordinates, which are non-linear. The implementation will be intensively linearised and the estimated uncertainty could be very wrong after a few transformations. If we use a Gaussian distribution in the polar coordinates, the uncertainty range will be relatively small and highly dependent on the centre that is initialised. In the tracking community, this problem was solved by introducing a parameterisation called modified polar (e.g., [128], [129]). For monocular SLAM, [24] proposed

Algorithm 10 The implementation of the proposed RB^2-PF visual odometry.

Initialise the particles (including the mean and covariance of the position, the mean and covariance of the velocity, and the particle weights), detect the visual features with the FAST corner detector [63] [64] and initialise the feature descriptors (using BRISK [25] in this chapter).

while Camera receives a frame **do**
  Perform camera state propagation (Section 4.3.1.2).
  Associate the features in the map with the ones on the image (Section 4.3.3.3).
  Sample the camera state (extended from Section 4.2.4.1 with modifications in Section 4.3.2.1).
  Perform camera (the Cartesian and angular) velocity update (similar to the idea in Section 4.2.4.2 with modifications in Section 4.3.2.1).
  Perform landmark location (in 3D space) update (extended from [110] and consider 4.3.1.3).
  Perform particle weight update (Section 4.2.4.3).
  Perform existence score update (Section 4.3.3.4).
  if There are not enough associated visual features **then**
    Detect new corners and initialise them as feature points (Section 4.3.3.2).
  end if
  if Effective sample size is low **then**
    Perform resampling (refer to [80]).
  end if
end while

(i)nverse (d)epth which is basically the idea of modified polar which enabled the EKF to address the Monocular SLAM problem. The landmark position is defined as follows:

$$L_{id}^{(i)} = \begin{bmatrix} x^{(i)} & y^{(i)} & z^{(i)} & \theta^{(i)} & \phi^{(i)} & \rho^{(i)} \end{bmatrix}^T, \tag{4.47}$$

where $\begin{bmatrix} x^{(i)} & y^{(i)} & z^{(i)} \end{bmatrix}^T$ is the position of the camera when the landmark is observed for the first time, and $\begin{bmatrix} \theta^{(i)} & \phi^{(i)} \end{bmatrix}^T$ is the orientation of the camera (represented using azimuth and elevation) when the landmark is observed for the first time. Both the position and orientation are coded in the global coordinates. In addition, ρ is the inverse depth.

The transformation function from the representation $L_{id}^{(i)}$ to (C)artesian space $L_C^{(i)}$ is defined as follows.

$$L_C^{(i)} = \begin{bmatrix} X^{(i)} \\ Y^{(i)} \\ Z^{(i)} \end{bmatrix} = \begin{bmatrix} x^{(i)} \\ y^{(i)} \\ z^{(i)} \end{bmatrix} + \frac{1}{\rho^{(i)}} \cdot m(\theta^{(i)}, \phi^{(i)}), \tag{4.48}$$

where $\left[X^{(i)}, Y^{(i)}, Z^{(i)}\right]^T$ is the Cartesian position of landmark, i, and

$$m(\theta^{(i)}, \phi^{(i)}) = \begin{bmatrix} cos(\phi^{(i)})sin(\theta^{(i)}) \\ -sin(\phi^{(i)}) \\ cos(\phi^{(i)})cos(\theta^{(i)}) \end{bmatrix}. \tag{4.49}$$

4.3.3.2 Landmark Initialisation

In common with other approaches (e.g., [112]), the input image is divided into a grid, and in each cell, the corner detector threshold is manipulated to ensure that a minimum number of corners are detected. The landmark points are extracted by a FAST corner detector [63] [64].[8] We calculate the BRISK feature descriptor [25] for each landmark in our system since it is relatively fast and precise [130]. For each landmark, we store an uncertainty of inverse depth (represented by a Gaussian distribution), an existence score, and a flag indicating whether it is a permanent map point. The subsequent subsections explain these stored quantities in more detail.

4.3.3.3 Landmark Matching

After predicting the camera state, all the landmark points will be assessed to check whether they could appear in the image and whether their feature descriptors are likely to be consistent to their initial value. The assessment contains the following criteria in order:

1. The existence score of the feature is higher than 0.4;
2. The angle between the current viewing ray and the original feature viewing ray should be smaller than 45 degrees;
3. The distance between the current camera position and the position in which the feature was first seen should be smaller than 30% of the feature depth from the position where the feature was first seen;
4. The feature-point position should project to a point inside the image.

This enables a subset of the landmarks to be obtained. For each landmark in this subset, the BRISK descriptors of its neighbours will be calculated. Then, we compare all the BRISK values of the neighbours with that of the landmark. The neighbour that is most similar to the feature of the landmark with a Hamming distance that is smaller than 0.15×512 will be matched (where 512 is the dimension of the BRISK descriptor). The neighbour size was identified empirically and optimised for each of the datasets.[9] To minimise the computational cost, we search in a smaller range and enlarge it if there is no match.

[8] https://www.edwardrosten.com/work/fast.html

[9] For example, we use a larger size, 9×9, and enlarge to 17×17 for TUM-RGBD because it contains many sudden moves, and we use size 7×7 for ICL-NUIM .

4.3.3.4 Existence Score

The quality of the feature points is evaluated using a probabilistic existence score inspired by a track initiation and deletion technique that was used in probabilistic data association (PDA) tracking (recall Section B.2. For more details, refer to [131]). The details of the implementation for this application will be described as follows.

Each landmark is initialised with a score of $P_{0|0} = 0.5$. Any landmarks with the score of less than 0.4 will be removed unless the depth uncertainty is small enough for it to be permanently preserved. Regarding the score prediction and update, assume that $P^{(k)}_{t-1|t-1}$ is the probability score of feature k at time $t-1$. At time t, the probability score prediction, $P^{(k)}_{t|t-1}$ (before receiving data at time t), is calculated as follows:

$$P^{(k)}_{t|t-1} = P_{22} \cdot P^{(k)}_{t-1|t-1} + P_{12} \cdot (1 - P^{(k)}_{t-1|t-1}), \tag{4.50}$$

where P_{22} denotes the probability that the target stays observable, and P_{12} is the probability that the target transfers to observable from unobservable. In this chapter, we use $P_{22} = 0.95$ and $P_{12} = 0.05$.

After receiving the landmark at time t, assuming $Z^{(k)}_t = \{z^{(k)}_{t,1}, ..., z^{(k)}_{t,N^{(k)}_t}\}$ is the set of associated detections for feature (k), we calculate:

$$\delta^{(k)}_t = \begin{cases} P_D P_G; & N_k = 0 \\ P_D P_G \left[1 - \frac{V^{(k)}_{G_t}}{\hat{N}^{(k)}_t} \cdot \sum_{i=1}^{N^{(k)}_t} \Lambda^{(k)}_{t,i}\right]; & otherwise \end{cases} \tag{4.51}$$

where P_D is the probability of detection (which is 0.75 in the experiment), P_G is the probability that the detection is inside the gate (which is 0.995 in the experiment), and V_G is defined as:

$$V^{(k)}_{G_t} = \pi \cdot \sqrt{|S^{(k)}_t|} \cdot G, \tag{4.52}$$

where $S^{(k)}_t$ is the uncertainty of the position of the landmark on the image, which is calculated via a scaled unscented transformation, given the camera state uncertainty and (static) landmark uncertainty. In addition, $G = 9.21$ corresponds to the value of P_G. We also assume that:

$$\hat{N}^{(k)}_t = N^{(k)}_t - P_D P_G P^{(k)}_{t|t-1}, \tag{4.53}$$

$$\Lambda_{t,i}^{(k)} = \frac{1}{P_G} \cdot \underbrace{\frac{1}{(2\pi)^{M/2}\sqrt{|S_t^{(k)}|}}}_{\mathcal{N}(z_{t,i}^{(k)};h(\begin{bmatrix} s_t^{pf} \\ O_t^q \end{bmatrix},\theta_t^{(k)});S_t^{(k)})} \cdot e^{-(d_{t,i}^{(k)})^2/2}, \tag{4.54}$$

where $z_{t,i}^{(k)}$ is the position of detection i and $h(\begin{bmatrix} s_t^{pf} \\ O_t^q \end{bmatrix}, \theta_t^{(k)})$ is the expected position of the landmark k on the image.

Finally, (4.55) is used to update the existence score:

$$P_t^{(k)} = \frac{1-\delta_t^{(k)}}{1-\delta_t^{(k)} \cdot P_{t|t-1}^{(k)}} \cdot P_{t|t-1}^{(k)}. \tag{4.55}$$

As the method that we use for matching landmarks with detections on the image, there is, at most, one detection for each landmark. As a result, $N_t^{(k)}$ equals either 1 or 0.

4.3.4 Experiments with RB^2-PF for Monocular Visual Odometry

The proposed approach will be tested with two public benchmarks: TUM-RGBD [2] and ICL-NUIM [3]. All the results of RB^2-PF and RB-PF are the median value over 10 executions, and the results from ORB-SLAM [112], LSD-SLAM [114], and SVO [132] either come from related papers or from using public release codes.[10].

'Freiburg3' of TUM-RGBD consists of several rectified image sequences captured by a hand-held Kinect device[11]. The camera is calibrated, and the parameters are provided. The videos include blurred frames, and there are also sudden rapid movements that is challenging to associate the landmarks with the image features and result in a larger uncertainty in the motion estimation. While choosing the test sequences, we consider both cases that the trajectory of the camera does and does not include loops, although the proposed algorithm does not use any kind of loop closure method. The name of the chosen videos and their basic information are presented in Table 4.5. Example images are shown in Figures 4.5 (a)-(d).

'Living Room (with noise)' in ICL-NUIM is a set of synthetic image sequences that is taken by a camera moving in a room (Examples are shown in Figures 4.5 (e)-(f)). The indoor environment contains desks, chairs, decorations, and flat walls. Although it is synthetic, photometric noises were added to simulate real camera input. The main difference from 'Freiburg3' is the better image quality (e.g., no blurring). However, the camera motion contains larger rotations and translations, which lead to greater motion ambiguity. There are more frames with less textures, which is quite challenging for the corner feature based SLAM/VO methods. While one

[10] https://github.com/raulmur/ORB_SLAM2 and https://github.com/tum-vision/lsd_slam
[11] https://developer.microsoft.com/en-us/windows/kinect

Video Name	Average Translational Velocity [cm/s]	Average Rotational Velocity [deg/s]	Duration [s]
sit_halfsphere	18.0	19.1	37.16
sit_tex_far	19.3	4.3	31.56
str_notex_far	16.6	4.0	27.29
sit_rpy	4.2	23.8	27.49
sit_xyz	13.2	3.56	42.51

TABLE 4.5: The average translational and rotational velocity of the tested videos in TUM-RGBD dataset [2].

could reduce the corner detection threshold, doing so causes numerous outliers to be generated. We use 3 out of 4 sequences ('lr_kt0n' – 'lr_kt2n') in the benchmark because. The reason for not using 'lr_kt3n' is that for a long period in this sequence, the corner detector can hardly detect enough features so the estimated trajectory diverges.

In the subsequent experiments, the estimated trajectories are aligned with the ground-truth. The scale is configured by the value when it can produce the minimum error. All the algorithms are evaluated by the absolute trajectory error (ATE) and the relative pose error (RPE). The description of these errors are in [2]. The RMSEs of the ATE and RPE are calculated using the public tool[12].

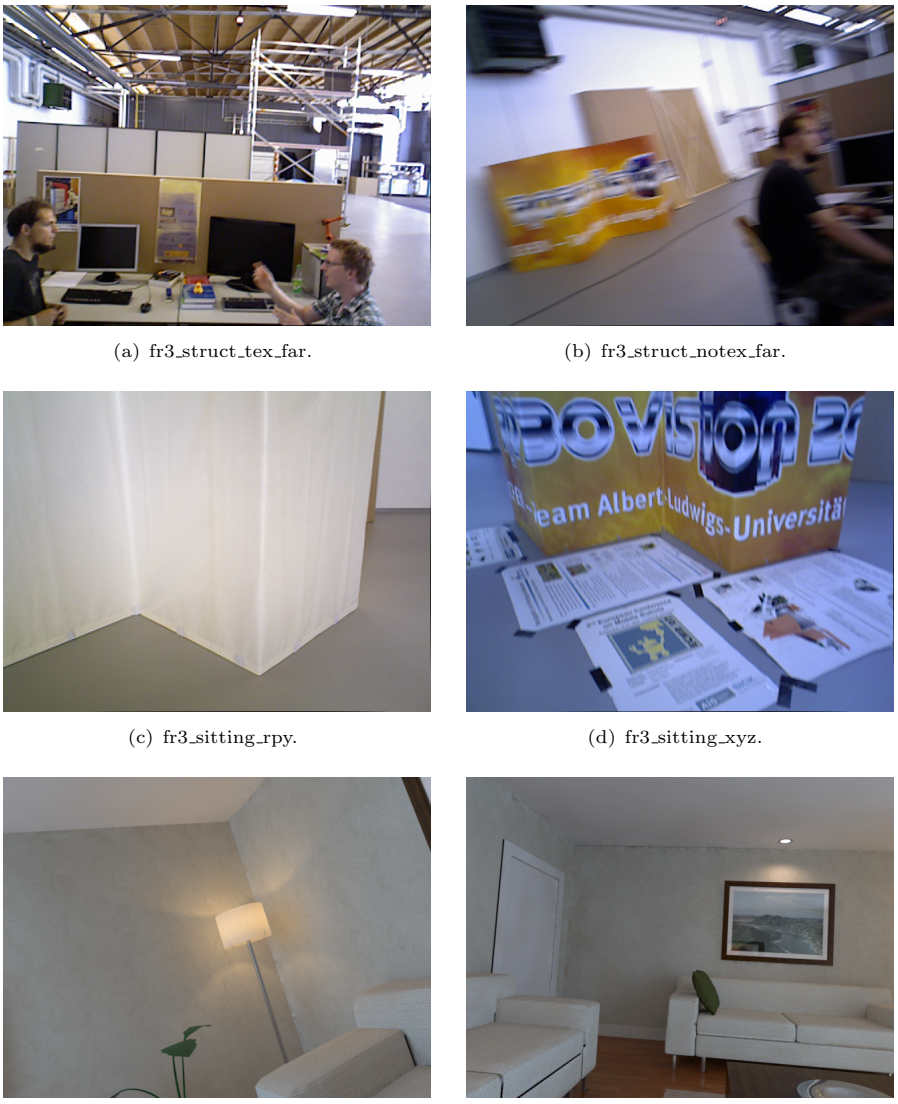
4.3.4.1 System Parameters

To see the generality of the system, we test it using different datasets with almost identical configurations[13]. The parameters for the filter are configured as follows:

- Frame rate: 30 Hz.
- Process noise: $\sigma_{pos} = 2.5^2\ m^2$ and $\sigma_{ori} = 1^2\ rad^2$.
- Measurement noise for each landmark: $R_t = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}^2$.
- Number of particles: 10, 20, or 30.
- The threshold for resampling: 0.5 ($\times$ 10, 20 or 30).
- Initial inverse depth: 0.2 m with variance $0.1^2\ m^2$.

[12] https://vision.in.tum.de/data/datasets/rgbd-dataset/tools

[13] My own experiments showed that particular parameters can increase the performance for different datasets. For 'fr3_sitting_rpy', due to the video contains intensive rotation and almost no translation, σ_{pos} is limited to 0.5^2 for meaningful results.



(a) fr3_struct_tex_far. (b) fr3_struct_notex_far.

(c) fr3_sitting_rpy. (d) fr3_sitting_xyz.

(e) Living Room kt0. (f) Living Room kt2.

FIGURE 4.5: Image examples from TUM-RGBD and ICL-NUIM benchmarks.

4.3.4.2 Comparisons between RB^2-PF and RB-PF

The implementations of RB^2-PF based and RB-PF based VO are modular with many common modules, such that the result only depends on the filter model used. In this subsection, both algorithms are tested on the TUM-RGBD benchmark. Table 4.6 and 4.7 show the RPE for both approaches. Figures 4.6–4.9 display the trajectories and the features that align to the ground-truth from RB^2-PF, ground-truth and the permanent feature points.

Video 'sitting_halfsphere' contains both intensive translations and rotations (see Table 4.5). The RPE of estimations from RB^2-PF is nearly 6 cm/s and 1.2 degree/s, which is not ideal but satisfying. By inspecting the estimated trajectory (Figure 4.6) with the video, the mistakes are

mostly occurred when the velocity of the camera changes rapidly (i.e., the blurred frames). Less matched features dramatically increase the uncertainty of camera state and sometimes influence the feature prediction in the following frames. This is one of the main reasons that the pose estimations are faulty. However, RB-PF is totally failed with very high translational errors which nearly equal to the average translational velocity. The minimum error among the 10 executions remains big which rules out the influence of randomness.

In the video 'sitting_xyz', the scene is similar to 'sitting_halfsphere', but there are less rotations and smaller average velocity on the translations. With 10 particles, RB-PF has good results which is approximating to RB^2-PF especially as many successful estimations. The reason is that there are relatively few blurred frame in this video which directly reduced uncertainties. However, when more than 20 particles are used, the improvement of RB-PF is not obvious. RB^2-PF outperforms the RB-PF on both translational and rotational pose estimation. RB^2-PF's trajectory is presented in Figure 4.7.

Next, 'sitting_rpy' is challenging and contains only rapid rotations. Since the camera motion does not include obvious translations during the whole time, no parallax is involved in this sequence. Therefore, the feature position cannot be estimated. Both approaches failed on estimating the true state of translations that both detect obvious translational movement. However, RB^2-PF is still better than RB-PF. In terms of the rotations, RB-PF can finally produce some successful estimations and the RMSEs are acceptable considering the very fast rotations of the camera. The improvement of using RB^2-PF remains outstanding.

The camera trajectory in the sequence 'str_tex_far' contains intensive translation and mild rotations which is kind of opposite to 'sitting_rpy'. unlike the above videos, the depth of the scene is not complex (see the feature points in Figure 4.8). For RB^2-PF, the translational and rotational errors are both smaller comparing to the videos above and the all the estimations are considered to be successful. Looking at RB-PF, the translational error is still not enough low. The rotational pose error is three times worse than the proposed RB^2-PF.

Generally speaking, it is obvious that RB^2-PF outperforms RB-PF on all the datasets. The conclusion is similar to what has been made for 2D SLAM. However, the difference between RB^2-PF and RB-PF is noticeably huge. RB^2-PF based VO can have more than 7 satisfying estimations over the 10 executions with even 10 particles, but RB-PF based VO can hardly produce more than 5 meaningful estimations with 30 particles. Increasing the number of particles can boost both algorithms, but the improvement for RB^2-PF is limited. It is because the performance is highly relied on the matched features, but the blurred frames cannot provide enough features for the particle filter. There has to be some randomness for those frames which accumulate after large number of iterations. The tables indication that 30 particles are not enough for RB-PF, however, due to the huge computational complexity, it is difficult to conduct the experiments with very large number of particles. For real VO applications, the RB-PF is likely to be prohibitive.

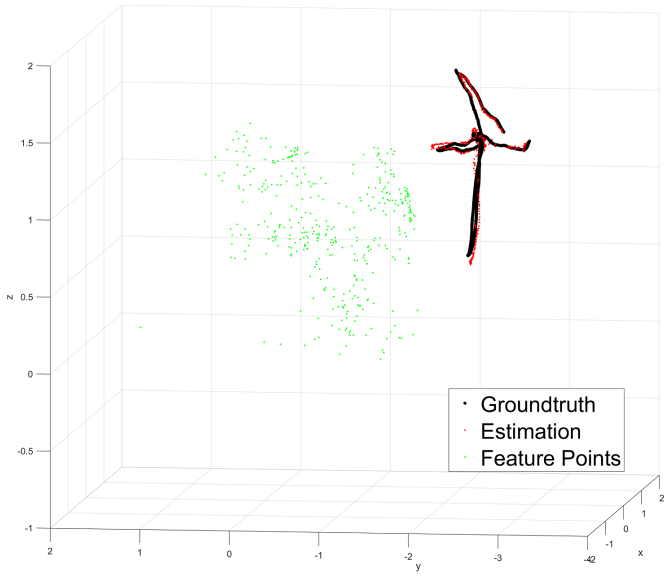


FIGURE 4.6: The trajectory estimated by RB^2-PF visual odometry vs ground-truth on dataset 'fr3_sitting_halfsphere'.

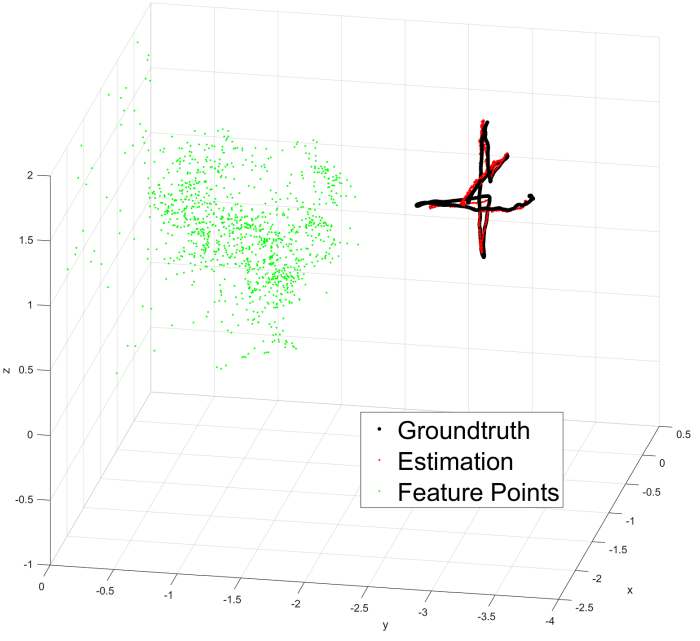


FIGURE 4.7: The trajectory estimated by RB^2-PF visual odometry vs ground-truth on dataset 'fr3_sitting_xyz'.

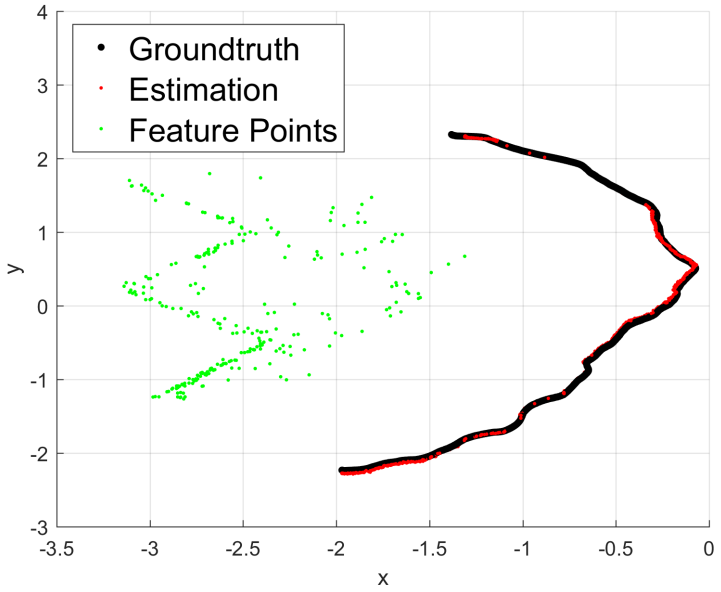


FIGURE 4.8: The trajectory estimated by RB^2-PF visual odometry vs ground-truth on dataset 'fr3_structure_texture_far'.

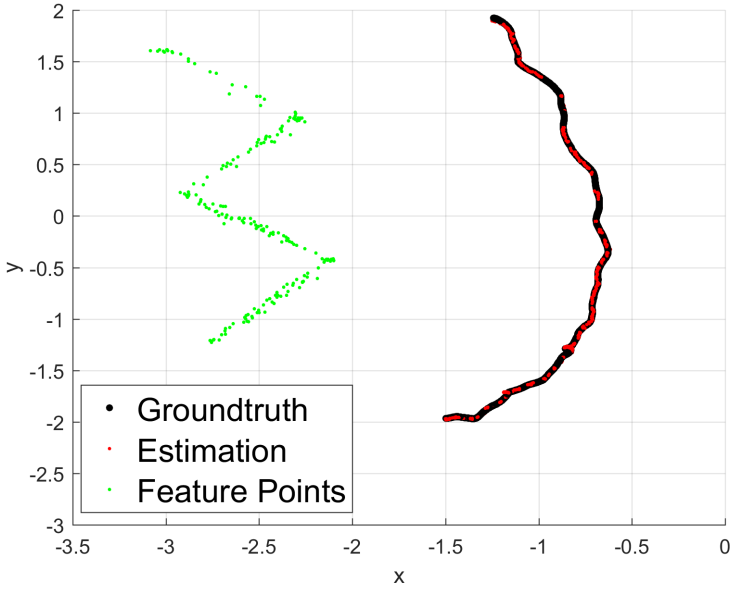


FIGURE 4.9: The trajectory estimated by RB^2-PF visual odometry vs ground-truth on dataset 'fr3_structure_notexture_far'.

Num. particles	RB^2-PF			RB-PF		
	sitting_halfsphere, translational error					
	success	min	median	success	min	median
10	7	4.56	6.18	0	11.43	20.51
20	7	4.09	7.30	0	14.84	18.61
30	10	4.04	5.95	0	13.52	15.85
	sitting_xyz, translational error					
	success	min	median	success	min	median
10	7	2.76	7.2	8	3.69	6.93
20	8	2.31	5.96	8	4.66	6.69
30	8	2.37	4.86	7	5.3	7.01
	sitting_rpy, translational error					
	success	min	median	success	min	median
10	10	2.87	4.57	10	4.57	4.63
20	10	3.4	4.23	10	4.56	4.63
30	10	2.79	4.08	10	4.51	4.61
	structure_texture_far, translational error					
	success	min	median	success	min	median
10	10	1.44	1.93	2	7.31	14.66
20	10	1.41	1.91	4	6.22	13.84
30	10	1.40	1.83	3	7.65	14.50

10 executions are made for each video using both approaches.
"success" means the translational relative pose error is smaller than 10 cm/s.

TABLE 4.6: The median and minimum relative pose errors (translation) using RB^2-PF and RB-PF on TUM-RGBD [2].

4.3.4.3 Comparisons of RB^2-PF to other algorithms

The previous section has proved that RB^2-PF is much better than RB-PF. In this subsection, the proposed algorithm will be compared with the state-of-the-art SLAM or VO algorithms. The testing benchmarks will be TUM-RGBD and ICL-NUIM. Table 4.8 reports the absolute trajectory RMSEs of the proposed algorithm, popular ORB-SLAM [112], LSD-SLAM [114], and SVO [132].

The first discussion is on the TUM-RGBD benchmark. In the video 'sitting_halfsphere' and 'sitting_xyz', the camera comes back to a similar position several times (see Figure 4.6 and 4.7), which is a big advantage for the algorithms with loop closure. The loop closure can sharply reduce the error by adjust the camera states off-line when revisiting is detected. Thus, ORB-SLAM unsurprisingly achieved the best estimation. The proposed RB^2-PF performs fine, though the video includes many blurred frames.

'sit_rpy' is a different scenario that the camera motion does not include translation, ORB-SLAM therefore could not initialise the position of features via triangulating; thus, there were

	sitting_halfsphere, rotational error					
	success	min	median	success	min	median
10	8	0.91	1.65	0	3.89	17.65
20	7	0.90	1.29	0	3.69	5.09
30	10	0.92	1.20	0	3.92	4.95
	sitting_xyz, rotational error					
	success	min	median	success	min	median
10	8	1.18	1.66	10	1.09	1.45
20	8	1.06	1.54	9	1.09	1.74
30	8	0.72	1.34	10	1.40	1.57
	sitting_rpy, rotational error					
	success	min	median	success	min	median
10	10	1.6	2.03	5	1.93	3.07
20	10	1.57	1.89	3	2.15	3.89
30	10	1.11	1.92	6	1.86	2.76
	structure_texture_far, rotational error					
	success	min	median	success	min	median
10	10	0.51	0.58	4	2.01	3.28
20	10	0.50	0.57	5	1.68	3.21
30	10	0.53	0.59	5	1.63	3.13

Ten executions are made for each video using both approaches.
"success" means the rotaional relative pose error is smaller than 3 deg/s.

TABLE 4.7: The median and minimum relative pose errors (rotation) using RB^2-PF and RB-PF on TUM-RGBD [2].

no subsequent camera states. The LSD-SLAM worked at the beginning, but it lost track due to intensive rotations, which come with blurred frames. The proposed algorithm outputs sensible rotation estimation. Both this sequence and 'str_nostr_tex_far' show the high robustness of the proposed idea. If the knowledge of the camera dynamics can be further obtained, a better estimation can be made accordingly.

In terms of the video 'str_tex_far', the case of revisiting (as shown in Figures 4.8) is not involved. Thus, the function of the loop closure in ORB-SLAM is not activated. The results show that ORB-SLAM outputs the best camera trajectories, but the proposed algorithm is better than LSD-SLAM. The best estimation (with minimum error) is approximating to ORB-SLAM's result. The reason that the proposal cannot beat ORB-SLAM is that the proposed approach is posing the problem as a filtering problem; thus, if there is a blurred frame where only a few features are matched, the uncertainty will increase dramatically. At this time, the uncertainty was heavily dependent on the pre-defined process noise (whose real value is unknown without additional sensors, such that a relatively large number was applied); therefore, the current uncertainty would be large. Thus, the uncertainty accumulates, and the filter-based algorithm will be worse

after a few more iterations. In contrast, as a keyframe-based approach, ORB-SLAM and LSD-SLAM can skip the blurred images and apply the optimisation over the history and thereby easily handle this situation. However, the disadvantage is that the keyframe-based approach cannot output a accurate real-time estimation (the refined estimation is provided after bundle adjustment) or the associated uncertainty. It is also challenging to sensibly fuse multiple-sensor data for these, especially when the sensors have a greater sampling rate than the frame rate.

Video 'str_notex_far' is textureless (see Figure 4.5-(c)). With a rather low corner detection threshold, there will be many outliers for feature matching. The ORB-SLAM could not find enough matched features for initialising the landmarks, as it uses RANSAC to eliminate the outliers. As a direct approach, LSD-SLAM worked well during the first 370 frames but lost track (for a total of 814 frames) after an intensive rotation. By decreasing the corner detection threshold and a well tuned process noise, the proposed algorithm finished the entire sequence (trajectory shown in Figure 4.9). It is worth highlighting that only corner features are considered in the proposed approach, and numerous features are on the edges. This becomes very challenging since the BRISK descriptor of a feature on an edge is not going to be outstanding. Our idea shows power when addressing very noisy inputs on monocular SLAM problems.

In terms of the ICL-NUIM benchmark, it is different from the TUM-RGBD mainly because there are no blurred frames. However, there are stronger rotations. It can show the results from another aspect. The RPE are shown in the second half in Table 4.8. The estimated trajectory, ground-truth, and landmarks from the sequence 'lr_kt0n'-'lr_kt2n' are shown in Figures 4.10-4.12.

Furthermore, 'lr_kt0n' contains camera revisiting and textureless frames. The results show that ORB-SLAM had the lowest error, but it benefited from the re-localisation, and only the position of keyframes were considered. The algorithm lost track for hundreds of frames in the middle, but re-localised successfully in the end. The SVO did a good job in this sequence, as it adopted an edge feature that was very helpful to the textureless frames. Our algorithm performed worst, as it only considered the corner features. However, it did not lose track, and the estimated trajectory was still roughly overlapping the ground-truth. Note that the re-identification is not available in our approach.

In the sequence 'lr_kt1n', most of the frames contain rich texture, which makes for a fairer comparison between SVO and RB^2-PF. The challenge includes the intensive rotation relative to the translation, but the translation is still enough for triangulating the landmark positions (such as Figure 4.11). From the table, we can see that RB^2-PF outperformed SVO. It shows that RB^2-PF is better than the optimisation-based approaches while estimating complex camera states. Although the textures are obvious in the images, and the camera translations are also obvious, ORB-SLAM failed on initialisation. The reason is likely the intensive rotation, which increased the ambiguity during triangulating.

Next, 'lr_kt2n' contains rich texture frames and the camera revisits. The ORB-SLAM performed best, due to the implementation of the loop closure and full bundle adjustment. The results of 'lr_kt2n' still shows the superiority of RB^2-PF relative to the optimisation-based estimation method (the trajectory of RB^2-PF is shown in Figure 4.12).

TUM-RGBD Freiburg3	RB^2-PF	ORB-SLAM	LSD-SLAM
sitting_halfsphere	5.8 (3.9)	1.34	5.87
sitting_xyz	7.5 (5.5)	0.79	7.73
sitting_rpy	4.9 (2.5)	×	×
str_tex_far	4.9 (2.1)	0.77	7.95
str_notex_far	7.8 (7.2)	×	×

For RB^2-PF, the minimum error over ten tests is displayed in the brackets.
"×" means the algorithm cannot initialise or lost track on more than 50% of the total frames.
Results of ORB-SLAM and LSD-SLAM on 'str_notex_far' and 'sit_rpy' are from my own execution using public code, different configurations for ORB-SLAM are tested for the best result. The remaining results are from [112].

ICL-NUIM Living Room	RB^2-PF	ORB-SLAM	SVO
lr_kt0n	18.20 (16.40)	0.25(Lost)	2
lr_kt1n	5.01 (3.36)	×	7
lr_kt2n	8.87 (5.32)	3.97	10

The results of SVO are from [132]. The results of ORB-SLAM are from my own execution using public code.

TABLE 4.8: Localisation error among 10 executions for RMSE of absolute trajectory error on TUM-RGBD [2] and ICL-NUIM [3] benchmarks.

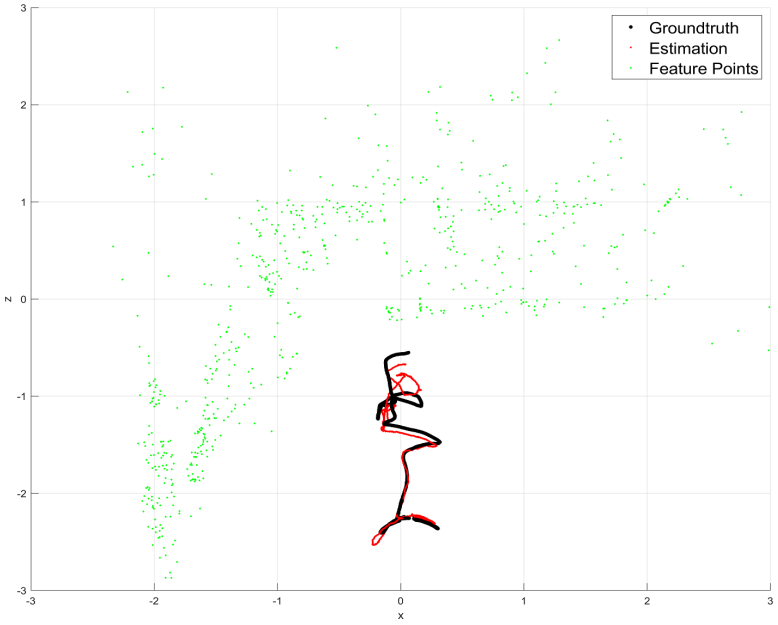


FIGURE 4.10: The trajectory estimated by RB^2-PF visual odometry vs ground-truth on dataset 'lr_kt0n'.

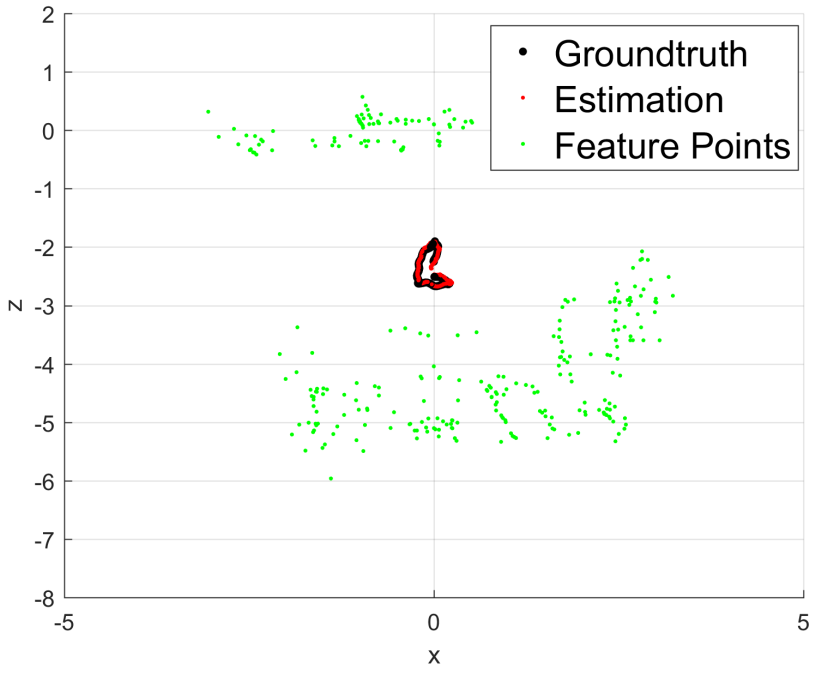


FIGURE 4.11: The trajectory estimated by RB^2-PF visual odometry vs ground-truth on dataset 'lr_kt1n'.

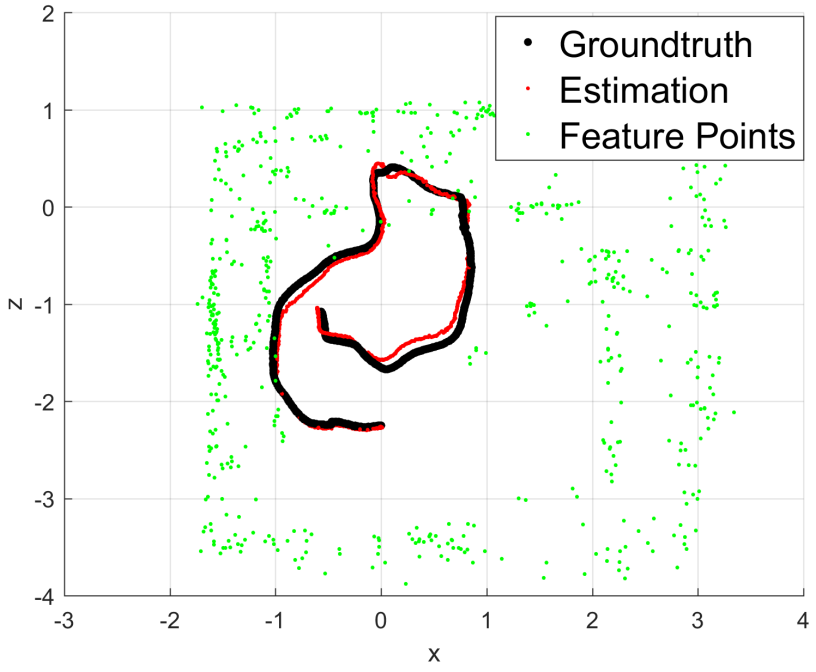


FIGURE 4.12: The trajectory estimated by RB^2-PF visual odometry vs ground-truth on dataset 'lr_kt2n'.

4.4 Conclusions

FastSLAM 2.0 [22] combines the idea of near-optimal proposal for the state-space and Rao-Blackwellisation for the landmarks in two dimensional SLAM problems. It is mentioned in [119] that the state-space can be further separated and the velocity can also be estimated by an analytic filter (i.e., Kalman filter). Using the achievements in Chapter 3, we propose RB^2-PF for solving visual odometry problems with monocular camera. Experiments indicate that the proposed algorithm offers improved performance in the context of realistic benchmarks relative to one of the best pre-existing FastSLAM 2.0 based approaches. Our system is also competitive compared to the state-of-the-art visual SLAM frameworks and does so with no need for either loop closure or bundle adjustment while considering some limited cases.

Chapter 5

Background Subtraction for a Moving Camera with Pronounced Parallax

5.1 Introduction

This chapter considers a more complicated but realistic situation than that in Chapter 2. The motivation is to detect moving objects in a video of a city that is obtained by a drone from a sufficiently low altitude in which the pronounced parallax is observed. As mentioned in Chapter 2, both the affine and homography transformation assume the scene to be flat. As such, the moving object detection algorithms that are based on both transformations do not perform well in the context of our motivating scenario where the three-dimensionality of the background needs to be considered. The objective of this chapter is to address the presence of the parallax, which generates large numbers of false alarms. Approaches do exist (e.g., [133]) that consider a background comprising multiple layers, as observed from a free-moving camera. They used RANSAC to estimate the homography matrices between two consecutive frames for each layer. They showed that it was possible to minimise the disparity between the rectified versions of consecutive frames using this idea. This idea has been considered in the context of moving object detection with background subtraction (e.g., [134, 135]).

In this chapter, we combine monocular visual odometry (VO) with background subtraction and motion compensation to detect moving objects from an airborne moving camera in the presence of a pronounced parallax. We use the position of the landmarks that are estimated by VO to calculate the multiple planes in the image. Motion compensation will be based on multiple planes. Thus, the compensated background model can be more precise than the approach that considers all the objects on a single plane. After the background subtraction, we project the detections to a unified 2D space (which is usually the ground plane of the video) such that a common dynamic model can be used for modelling target moves, and the visualisation can be friendlier to the users. Finally, the detections are tracked using a Gaussian mixture probabilistic hypothesis density (GM-PHD) filter (as described in [77]).

The rest of the chapter is organised as follows. Section 5.2 describes some existing algorithms that are adopted in the proposed system. Section 5.3 describes how multiple planes are estimated

and used to handle the pronounced parallax. This section will also describe how we project the detections from the image to a unified space. Section 5.4 includes two experiments involving a simulated video of a city and real video from an outdoor benchmark.

5.2 Related Work

5.2.1 Monocular Visual Odometry

Monocular simultaneous localisation and mapping (SLAM) and VO are similar at some points. Although advanced SLAM algorithms include traversing historic data to correct previous errors (i.e., loop closure), we are only interested in the real-time estimation of the camera state and the positions of the landmarks.[1] In the proposed system, we can use the output from either a SLAM or VO algorithm, such as ORB-SLAM [112] or SVO [132], respectively (assuming that they are configured to work well on the input video). In our experiments, we choose to use a recently developed VO solution [16], RB^2-PF based VO, primarily because it works well on both videos we consider in our experiments.

Recall that (as in Section 4.3.1) the camera states are defined by:

$$s_t = \begin{bmatrix} v_t & o_t & \dot{v}_t & \dot{o}_t \end{bmatrix}^T, \tag{5.1}$$

where v_t is the Cartesian position of the camera, o_t is the orientation of the camera (represented using quaternions), $\dot{v}_t$ is the Cartesian velocity, and $\dot{o}_t$ is the orientation velocity (represented using Euler angles).

We consider a monocular pin-hole camera model without distortion. The inverse depth [24] is used to parameterise the position of landmark. Note that, due to the geometry of a monocular camera system, it is impossible to estimate the exact distance between an object and the camera (i.e., the distances between the estimated positions of any two landmarks are scaled based on an unknown ratio). As a result, the projected detections and the resulting estimated trajectories are all scaled relative to reality.

5.2.2 Background Subtraction

In theory, most of the background subtraction algorithms (e.g., [20, 136]) should be compatible with the proposed system. In the cases of interest herein, the background content is often rapidly changing (due to camera motion and parallax). To maintain a low computational cost, we use a codebook-based background subtraction algorithm proposed in [26], which is briefly described as follows. For each pixel, the background model has K background templates, $\{T_1, T_2, ..., T_K\}$, and every template includes a pixel value that is represented using the YC_bC_r colour space. The

[1] The off-line correction does not affect the accuracy in our system, as we only care about the current output. That said, our system should be able to use most of the existing feature-based SLAM algorithms.

pixel value is also associated with a counter. The background classification process compares an input pixel value with all the background templates. If the difference between any template and the input is lower than a set of thresholds, $\{\varepsilon^Y, \varepsilon^{C_r}$, and $\varepsilon^{C_b}\}$ on all the three colour channels, then the pixel is classified as background. Afterwards, its counter will increase by one, while the counters of other templates will decrease by one. If any counter for a background template decreases to zero, the template will be deactivated and will no longer be involved in the future processes until it is initialised again.

To add new background templates, another template T_A is involved (and has the same data structure as the other background templates). As before, if this template matches the foreground pixel value, its counter increases by one. If it does not match the current pixel, its counter decreases by one. When its counter equals zero, the pixel value for this template is substituted with the current pixel value. If its counter is greater than θ_A, T_A is considered a new background candidate.

This algorithm has been designed to process long-term video from a static camera involving fewer changes within the background than are considered here. The full version of this approach therefore considers several more functions for gradually updating the templates, detecting false alarms, and adapting the thresholds. However, these functions are not needed, and thus are not implemented here. For more details, the interested reader can refer to [26].

5.3 Moving Background Subtraction Against Parallax

5.3.1 Motion Estimation For Multiple Planes

The aim of motion estimation is to find the motion vector of each pixel between two consecutive frames, I_{t-1} and I_t, and to minimise the difference between the rectified I_{t-1} and I_t. Our approach is to consider the motion to be well approximated by assuming that the scene comprises multiple planes.[2] Since our motivation is to detect moving objects in urban areas, it is sensible to consider one of the multiple planes to be the ground plane, θ_0. This plane can therefore be used to determine the high parallax regions (e.g., high buildings) and the unified space to which all detected vehicles will be projected. To ensure that we estimate the planes accurately, we only use the landmarks from VO with a small inverse depth uncertainty. This ensures that the process of background subtraction will not run in the initial frames of a video until sufficiently many observations of each landmark are obtained.[3] Note that the considered landmarks include the points in the current frame and those that are not currently detected but which have historically been detected sufficiently often that their uncertainty in location is small. To limit the computational cost, a maximum number of landmarks to consider is configured.

[2]The existing algorithm assumes that the scene is on one plane. It will be called the 1-plane approach in the rest of this chapter.

[3]The time taken for enough information depends on the camera motion and the image texture. If the translational movement of the camera is pronounced and there are enough landmarks, it takes fewer frames to obtain enough satisfying landmarks. In our experiments, the city simulation (described in Section 5.4.1) takes 50 frames (two seconds) to obtain enough landmarks, and the outdoor video (described in Section 5.4.2) takes nearly 150 frames (nearly 6 seconds) due to less intensive camera movement.

With the assumption that, in an urban area, there are more landmarks on the ground plane than on the other planes, RANSAC is used to calculate a shared plane of those landmarks. The ground plane is represented by:

$$\theta_0 = \begin{bmatrix} a_0 & b_0 & c_0 & d_0 \end{bmatrix}, \tag{5.2}$$

which satisfies

$$a_0 X + b_0 Y + c_0 Z + d_0 = 0, \tag{5.3}$$

where $\begin{bmatrix} X & Y & Z \end{bmatrix}$ are the Cartesian coordinates of a point.

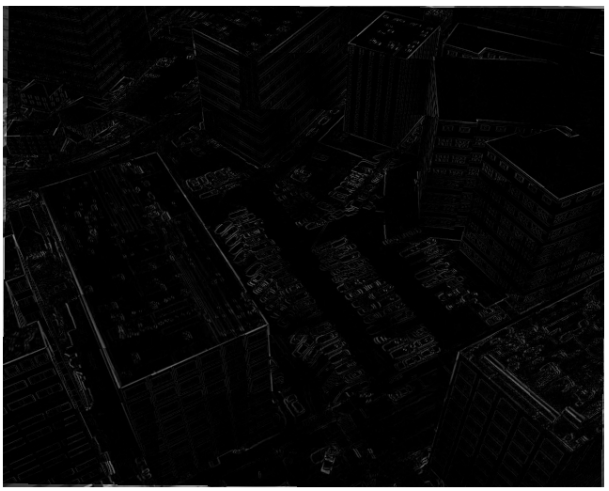
While RANSAC often works well, it is inevitable that the estimation fails to work well on one or two frames (e.g., due to the distribution of landmarks). It is possible that the ground plane needs to be maintained (e.g., because of changes in terrain or because of problems caused by the VO algorithm being fallible). To overcome this issue, we consider the same ethos as was adopted in the context of the background template. More specifically, we associate the ground plane and the candidate ground plane with counters. If the input plane is very different from the current ground plane, we still use the current ground plane but treat the input plane as a candidate. If the candidate plane appears often that its counter is larger than the threshold, the ground plane will be substituted by the candidate. While this scenario is not always necessary, in our experiment, it is useful to avoid system failure.

Once the ground plane is estimated, we apply RANSAC to all the landmarks on the current frame whose distance to the ground plane is larger than the threshold φ_d. This operation is used iteratively to find more planes, but we only admit the planes that have more than eight points on them. In our system, we consider three planes (including the ground plane), denoted by $\left\{ \theta_0, \quad \theta_1, \quad \theta_2 \right\}$. The choice of the number of the planes is empirical by observing the scene and the quantity of extracted landmarks. For some videos with rich content in the scene, more planes can be applied.

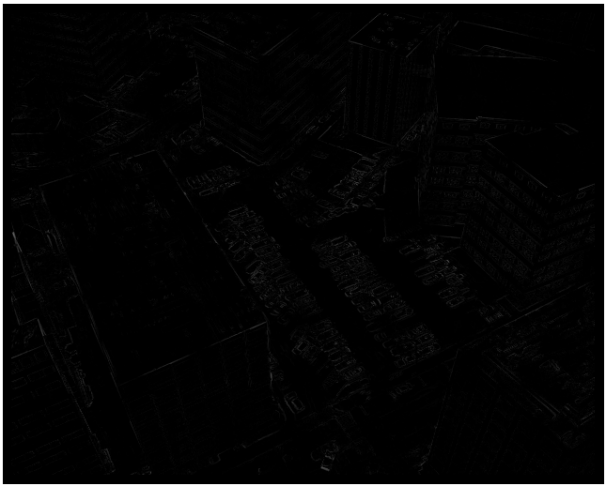
After the planes are estimated, it is necessary to estimate which landmarks are on each plane. To do so, for each plane, we estimate the global motion (in the form of a homography matrix) with Lucas-Kanade's method [71] using only the pixels around the landmarks on that plane. The homography matrices are $\left\{ \eta_0, \quad \eta_1, \quad \eta_2 \right\}$ corresponding to the planes $\left\{ \theta_0, \quad \theta_1, \quad \theta_2 \right\}$. Finally, for each pixel, we calculate the three potential motion vectors (one using each one of $\left\{ \eta_0, \quad \eta_1, \quad \eta_2 \right\}$) and the disparities of that pixel between I_t and the three rectified I_{t-1}. The final motion vector takes the one that leads to the smallest disparity. Figure 5.1 shows how the image disparity between the rectified I_{t-1} and I_t is reduced using this approach.

5.3.2 Estimate and Process Regions of Pronounced Changes

The three-dimensionality of the urban environment not only results in motions that are inconsistent with a global homography transformation but also gives rise to pixels that, because of the change in viewpoint, are obscured and become visible (see Figure 5.2 for examples). To handle such pixels, our solution is to identify regions of the image where changes are pronounced and



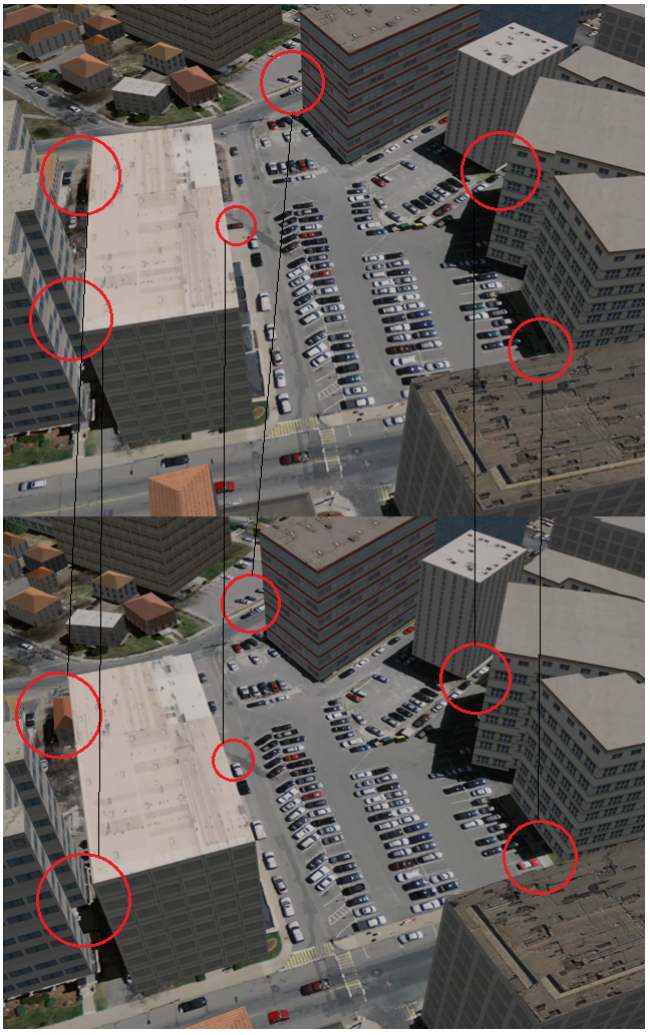
(a) Direct homography that considers everything to be on one plane.



(b) After merging the homography matrices of three planes.

FIGURE 5.1: An example of the disparities between rectified I_{t-1} and I_t using two motion estimation approaches on a city simulation dataset.

only use the most recent frames to construct the background in these regions. The regions where changes are pronounced are estimated by first finding the disparity between the rectified I_{t-1} (using the homography matrix of the ground plane η_0) and I_t. Then, a map is built based on the disparity by marking the pixels for which the difference is larger than a threshold, φ_i. The causes of those marked regions can be high parallax, moving objects, or errors in motion estimation. It is hard to distinguish between these causes directly in complex environments. However, since we are focused on the video of urban areas, we assume that the parallax is mainly introduced by high buildings (with straight edges). To ensure we do not erroneously infer that these regions are caused by moving objects, we find lines in the map using a Hough transform. The lines identified via this process are then widened and assumed to identify parts of the image where changes are caused by the three-dimensionality of the urban environment.



Top: frame index 500. Bottom: frame index 520. Due to the different perspectives, some objects are obscured which cannot be modeled by image warpping.

FIGURE 5.2: An example of content insertion due to parallax.

5.3.3 Background Subtraction

The initialisation of the background modelling copies each pixel in the initial frame to the background template and sets the associated counter to one. After each time the motion vectors are obtained (using the approach described in Section 5.3.1), the background model is compensated. The background subtraction and model update follow the basic procedures described in [26]. However, we make the following modifications:

- Parameter θ_A is set to be much smaller than in the original description (in [26]). This ensures that a new template takes less time to be considered a background candidate.
- In the regions where changes are pronounced (as identified in Section 5.3.2), if the pixel is foreground, the counter for its background template will be decreased by two rather than by one. If a pixel is background, the counter will be increased by two. This process ensures

that the updating is rapid in regions where pronounced changes occur. In these regions, content is consistently inserted, and recent changes should have a significant influence on the estimate of the background.

- The result of background subtraction excludes the regions where changes are identified as being pronounced. This approach can be helpful in terms of reducing the number of false alarms.[4]
- Image morphological operations are applied to remove noise.
- Since we are using a very small θ_A, the updating process for the template, T_A (as described in Section 5.2.2), will be based on the background subtraction result after excluding the regions of significant change and applying morphological operations. If a pixel is classified as background, the normal updating process is used. If a pixel is classified as foreground, it has a probability of 0.25 in the experiment to perform the normal updating process. For an ideal case, the pixels that are classified as foreground should never be updated. However, the background subtraction cannot be perfect for variational frames, and the probability tackles the bad cases.

5.3.4 Projecting Detections to the Unified Space

The objective of this chapter is to finally track the moving objects in the frames. However, considering a dynamic model for detection of movement in the image will be challenging when the camera moves and rotates. Our approach projects the detections to a unified (albeit arbitrarily scaled) space and offers a solution to this problem. With the help of the previously estimated ground plane, the projection can be calculated by solving equations (4.36), (4.37), and (5.3) and the details are as follows.

We first calculate the following parameters.

$$j = R_{11} - \frac{R_{13}a_0}{c_0} - R_{31}u + \frac{R_{33}a_0u}{c_0}, \tag{5.4}$$

$$p = R_{32}u - \frac{R_{33}b_0u}{c_0} - R_{12} + \frac{R_{13}b_0}{c_0}, \tag{5.5}$$

$$m = -R_{31}v_{t,x}u - R_{32}v_{t,y}u - R_{33}v_{t,z}u - \frac{R_{33}d_0u}{c_0} + R_{11}v_{t,x} + R_{12}v_{t,y} + R_{13}v_{t,z} + \frac{R_{13}d_0}{c_0}, \tag{5.6}$$

$$k = R_{21} - \frac{R_{23}a_0}{c_0} - R_{31}v + \frac{R_{33}a_0v}{c_0}, \tag{5.7}$$

$$q = R_{32}v - \frac{R_{33}b_0v}{c_0} - R_{22} + \frac{R_{23}b_0}{c_0}, \tag{5.8}$$

$$n = -R_{31}v_{t,x}v - R_{32}v_{t,y}v - R_{33}v_{t,z}v - \frac{R_{33}d_0v}{c_0} + R_{21}v_{t,x} + R_{22}v_{t,y} + R_{23}v_{t,z} + \frac{R_{23}d_0}{c_0}, \tag{5.9}$$

[4] It is possible that the moving objects are also removed when any moving objects are near the regions where pronounced changes occur. This problem could be solved by predicting the likely position of the object (using the dynamic model) and using appearance matching.

where $\begin{bmatrix} u_t^{(i)} & v_t^{(i)} \end{bmatrix}^T$ relates to the position of the target: (i), in the image at time t, which follows the pin-hole camera model (descried in Section 4.3.1.3). $\begin{bmatrix} v_{t,x} & v_{t,y} & v_{t,z} \end{bmatrix}^T$ is the Cartesian position of the camera, and $\begin{bmatrix} a_0 & b_0 & c_0 & b_0 \end{bmatrix}^T$ is the representation of the ground plane. Moreover, we define the following equation:

$$R(o_t) = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}, \tag{5.10}$$

where $R(.)$ is the transition function from the quaternion orientation to rotation matrix, and o_t is the quaternion orientation of the camera at time t.

The position of the target in the global space, $\begin{bmatrix} X_t^{(i)} & Y_t^{(i)} & Z_t^{(i)} \end{bmatrix}^T$, is calculated as follows:

$$X_t^{(i)} = \frac{(p \cdot n - q \cdot m)}{(p \cdot k - q \cdot j)} \tag{5.11}$$

$$Y_t^{(i)} = \frac{(k \cdot m - j \cdot n)}{(j \cdot q - k \cdot p)} \tag{5.12}$$

$$Z_t^{(i)} = \frac{(-d_0 - a_0 \cdot X - b_0 \cdot Y)}{c_0} \tag{5.13}$$

Algorithm 11 Implementation Of Proposed Algorithm

Suppose we have 3D Cartesian position of current and history landmarks, background templates and current frame.

1: Calculate planes with landmark positions (Section 5.3.1).
2: Calculate the homography matrices between pixels on each plane and the background (Section 5.3.1).
3: Compensate the background with the homography matrices calculated for each of the planes.
4: Estimate the regions with pronounced changes (Section 5.3.2).
5: Perform background subtraction (Section 5.3.3).
6: Update the background with the current frame and regions with pronounced changes.

5.4 Experiments

The proposed moving object detection algorithm (see Algorithm 11) will be tested on two datasets. The first one is a simulated video of a city [137] and exemplifies the motivation for this chapter. The camera is at a low altitude and moving along a curved trajectory (with significant rotations). The parallax is pronounced. The moving objects are mainly cars. The

second dataset is one of the videos[5] provided in [1]. There is little parallax present in the video, which was recorded in an outdoor urban area by an unmanned aerial vehicle. This video is used to test whether the algorithm is extensible to other cases and applicable in real environments. Only the moving objects larger than 15×15 pixels are considered by either the detector or evaluation. If there are two detections for one object, only one is treated as the correct detection. If there is one detection for two nearby objects, it is treated as two correct detections. Finally, the detections are processed by a GM-PHD [77]) filter. Before applying the GM-PHD filter, the detections in the frame will be projected to a unified space (as discussed in Section 5.3.4). Thus, the detection are in the three dimensional space and a constant velocity model can be applied straightforwardly. Note that the GM-PHD filter is not meticulously optimised for this problem and the parameters are presented in Table 5.1.

For both experiments, we use identical sets of parameters as follows.

Before the visual odometry algorithm, we first define the camera calibration parameters [6]:

- The centre of the frame is assumed to be the principle point.
- The focal length is configured: $\begin{pmatrix} f_x, & f_y \end{pmatrix} = \begin{pmatrix} 400, & 400 \end{pmatrix}$.
- The frames are not distorted.

The parameters for the visual odometry system is configured as follows.

- 30 particles are used in the particle filter.
- The process noise: $\sigma_{pos} = 0.5^2\ m^2$ and $\sigma_{ori} = 0.5^2\ rad^2$.
- The measurement noise: $R_t = \begin{bmatrix} 2 & 2 \end{bmatrix}^2$.
- The initial inverse depth: $0.2\ m$ with covariance of $0.1^2\ m^2$ (for more descriptions, please refer to Section 4.3 or [16]).

For background subtraction, we use the following fixed thresholds:

- $K = 3$ and $\{\varepsilon^Y, \varepsilon^{C_r}, \varepsilon^{C_b}\} = \begin{Bmatrix} 12, & 6, & 6 \end{Bmatrix}$ (see Section 5.2.2).
- $\varphi_d = 0.25$ (see Section 5.3.1).
- $\varphi_i = 20$ (see Section 5.3.2).
- $\theta_A = 4$ (see Section 5.3.3).

Parameter	Value	Parameter	Value
d_t	1	η_{del}	0.01
ν_{init}	0.1	P_d	0.75
μ_{init}	$[0;0;0;0;0;0]^T$	Q	Equation (3.6) when $D = 3$ and $\sigma = 0.05$
Σ_{init}	$diag([0.1;0.1;0.1;0.05;0.05;0.05]^2)$	R	$diag([0.1, 0.1, 0.1]^2)$
η_{merge}	6	$n_{clutter}$	$poissrnd(5)$
$\eta_{display}$	0.75		

Target spawning is not considered in these detections.
Please refer to Table 2.5 to see the descriptions of the parameters

TABLE 5.1: The essential parameters for GM-PHD filter while tracking the detections as described in Section 5.4.1 and 5.4.2.

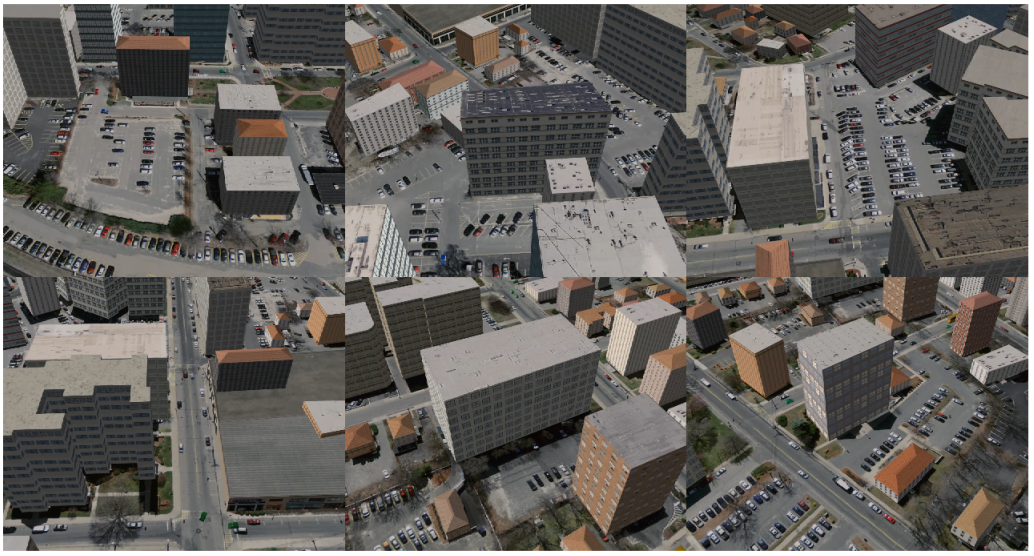


FIGURE 5.3: A few image examples of the video sequence of city simulation.

5.4.1 Test With City Simulation

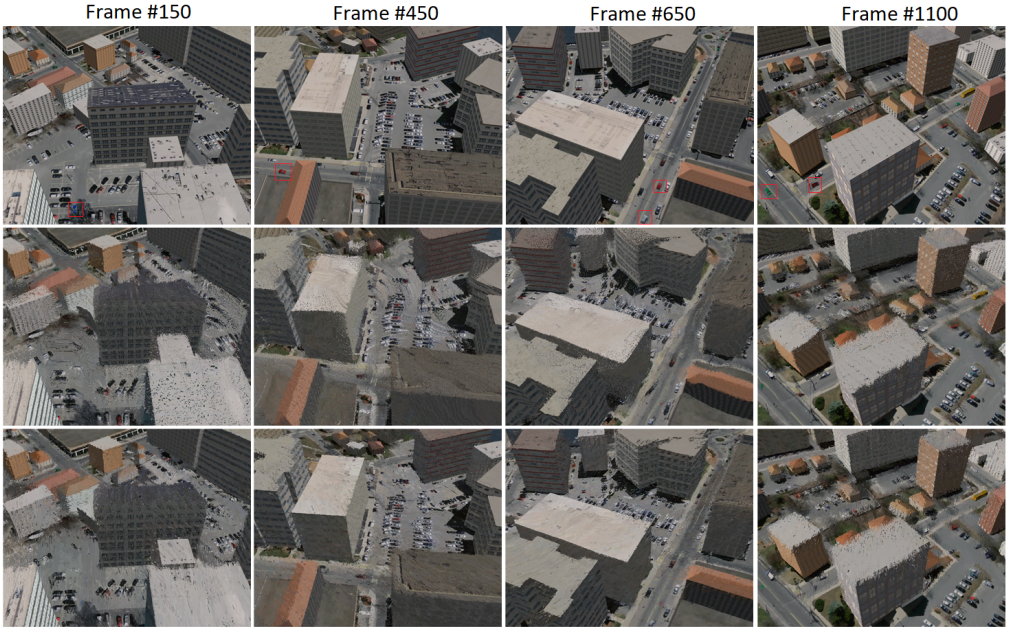
Some example frames in the sequence are shown in Figure 5.3. This sequence includes 1250 frames (each of 640 × 512 pixels). The first 50 frames are used for estimating the multiple planes; therefore, we only consider the moving objects from Frame 51 to Frame 1250. There are, in total, 1432 moving objects in this sequence period.

Figure 5.4 shows the estimated backgrounds using the existing approach ('1-plane method'), which considers all objects on one plane, and the proposed approach, which considers multiple

[5]https://ivul.kaust.edu.sa/Pages/Dataset-UAV123.aspx
[6]In the chapter, we do not use the true configuration of the camera. Our motivation is to identify whether an algorithm we have developed can be used on any surveillance videos (not on video from a specific camera).

planes for compensating the moving frames. As the video cannot be presented here, it is necessary to explain that only the vehicles in the red rectangles are moving objects. If we focus on the difference between the two estimated backgrounds and compare them with the background area of the input frame, it is evident that the estimation using the proposed idea is more similar to the background of the input frame. For example, in Frame 150, the roof of the bottom-right and bottom-left buildings should move downwards with faster speed relative to the ground. However, such disparity of speed cannot be reflected by considering only one plane and the related transformation matrix. We can see the obvious differences between the original frame and the estimation using the 1-plane method. Those differences of the building will be considered moving objects. In contrast, the background estimation using the proposed idea is better. Although the differences between the true background and the estimation are still observable, the small amounts of noise can be easily eliminated with some morphological thresholds.[7] Similar logic can be used to present the advantages of the proposed idea for Frames 450, 650, and 1100 in Figure 5.4.

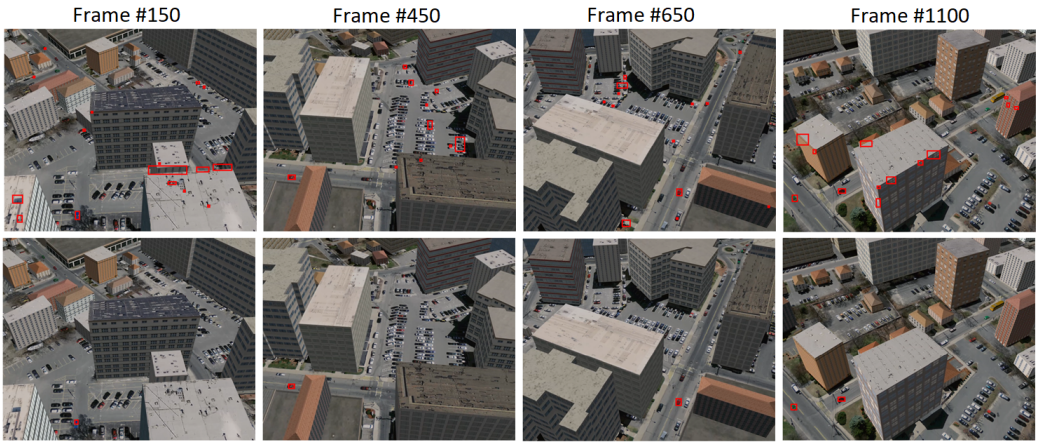
Figure 5.5 shows the detections using two approaches for the same frames mentioned above. Inspecting Figures 5.5 and 5.4, the false alarms mostly originate from an incorrect background estimation. The conclusion can be made that the existing approach cannot do well with pronounced parallax.



First row: original frame. Second row: estimated background using the 1-plane method. Third row: estimated background using the proposed approach.

FIGURE 5.4: Background estimations using the proposed approach and the approach considering one plane.

[7] The same thresholds are actually adopted by the existing approaches, but there are still numerous false alarms remaining.



First row: detections using the 1-plane method. Second row: detections using the proposed approach.

FIGURE 5.5: Detections using the proposed approach and the approach considering one plane.

Considering the complete sequence, the proposed moving object detector yields 2080 detections, and 1108 of them are correct. After filtering with the GM-PHD filter, the confirmed detections reduce to 1362, of which 1026 are correct. The number of moving objects detected by the existing algorithm is 13770, and 1316 of them are correct. After adopting a GM-PHD filter, there were 9414 confirmed detections, and 1248 of them are correct. The respective precision and recall are reported in Table 5.2.

By inspecting the precision in Table 5.2, it is obvious that the improvement from our proposed approach is huge with or without help from applying a GM-PHD filter. For the existing approach, as the false alarms are usually around the areas with pronounced parallax (stationary buildings or ground objects with rich textures), the probability of their detection cannot be modelled by a Gaussian distribution; thus, false alarms are difficult to filter out with the existing probability based filters. Comparing to only using the proposed detection approach, the contribution of applying a GM-PHD filter is apparent as well.

Although the presence of false alarms using the proposed approach is reduced, the remaining false positives are still primarily caused by faulty motion compensation. We use landmarks from VO to estimate multiple planes. Because of the condition that only the plane with more than eight landmarks on it is considered in the proposed motion compensation method, any pixels on a plane with a small number of landmarks can generate false negatives.

From the recall in Table 5.2, we can see both approaches suffered from missed detection. Since the camera orientation varies, when the angle between the camera and horizon is small, objects that are far away tend to move slowly in the image. As previously discussed, we must consider more recent frames to deal with the parallax and dynamic background. This makes the slowly moving objects hard to detect. The table also illustrates that the proposed approach has more missed detections (lower recall) than the existing one. When the frame motion is represented by several planes, the detected moving object could be either the background on one of the planes or an actual moving object on the ground plane. If the movement of the actual

moving object is similar to the motion of one plane, that moving object could be classified as background on that plane. When a GM-PHD filter is adopted for tracking, the recall reduces reasonably.

Number of Detections

Method	Precision	Recall
Multi-plane Background	0.533	0.774
1-plane Background	0.096	0.919

GM-PHD Refined Detections

Method	Precision	Recall
Multi-plane Background	0.753	0.716
1-plane Background	0.133	0.872

The multi-plane background is the proposed approach.
The 1-plane Background is similar to what was proposed in Chapter 2.

TABLE 5.2: The experiment result on the video of the simulated city using our approach.

Figure 5.6 presents several tracks from different sequences of frames. This figure shows the feasibility of successful tracking with the proposed algorithm and a multi-target tracker. The proposed algorithm works well from different aspects of views. This system can detect and track the objects even when the appearance of the object is obscured (e.g., the last frame in Figures 5.6 (a)(b)(e)(g)), which is an advantage of the background subtraction based approaches compared to the foreground detectors.

Figure 5.7 shows the confirmed tracks in the global space and a set of examples of their positions in the frames. This figure shows good quality of the projection from frame to the unified space, though intensive translations (e.g., Frame 200, 250, 300) and rotations (e.g., Frame 400, 450, 650) appear in the sequence. The tracks indicate the road network of the (simulated) city. By investigating the road network, it reflects some hidden facts that some roads on different frames are actually connected. For example, the crossroad in Frame 25 connects to the main road in Frames 200, 250, and 300. It is noticeable from the video, but not obvious. However, the fact that the main road in Frame 400 connects to the main road in Frame 945 is relatively difficult for human to spot after a few confusing rotations and translations. Therefore, this centralised visualisation can provide aid for monitoring moving objects on a large scale.

(a) Tracking results from Frame #7 to #72.

(b) Tracking results from Frame #8 to #105.

(c) Tracking results from Frame #195 to #245.

(d) Tracking results from Frame #227 to #30.

(e) Tracking results from Frame #380 to #462.

(f) Tracking results from Frame #1033 to #1078.

(g) Tracking results from Frame #1098 to #1165.

FIGURE 5.6: Selected tracking results using proposed detector and GM-PHD filter on the simulated city sequence.

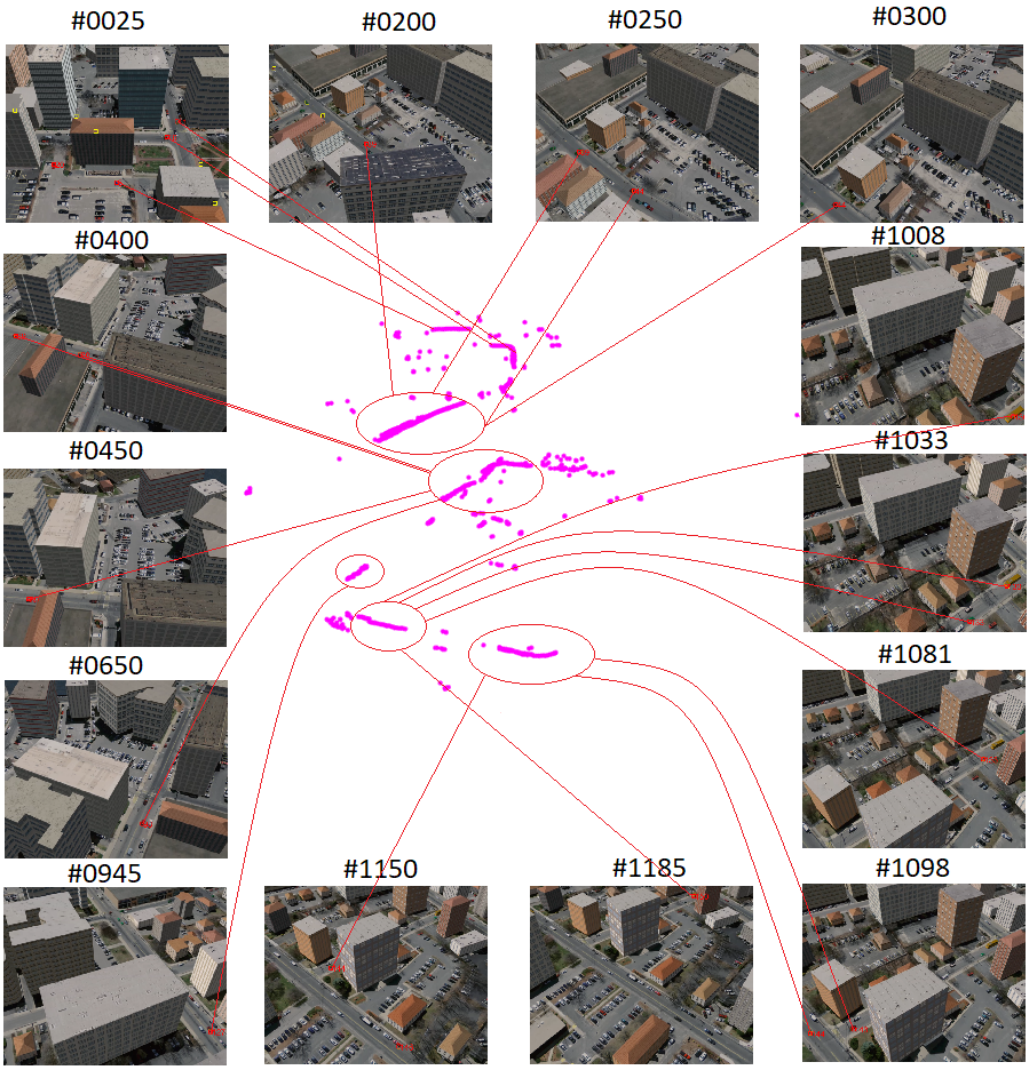


FIGURE 5.7: The positions of confirmed tracks in the global space and their positions in some frames.

5.4.2 Test with an Outdoor Benchmark

FIGURE 5.8: Example frames of the real outdoor benchmark [1].

Some example frames in the sequence (from the public benchmark [1]) are shown in Figure 5.8. This sequence includes 1650 frames (each of 640×512 pixels). The first 149 frames are used for estimating multiple planes; therefore, we only consider the moving objects from Frame 150 to Frame 1650. There are, in total, 2796 moving objects in this part of the video.

The proposed moving object detector yields 4103 detections, and 2472 of them are correct. After filtering using a GM-PHD filter, the total detections reduce to 3056, of which 2371 are correct. As a comparison, the number of moving objects detected by the existing algorithm (1-plane) is 5414, and 2514 of them are correct. After adopting a GM-PHD filter, there were 3833 detections, and 2447 of them are correct.

Table 5.3 shows the precision and recall on the outdoor benchmark. The precision of the 1-plane approach is much higher than that on the dataset of city simulation. As captured in a rural area, this sequence contains less parallax than the other dataset. The trees are the main objects that cause parallax; therefore, this algorithm may not consider multiple planes for motion compensation, as it always does in the other dataset. This is why the improvement for precision is not as strong as in the previous experiment. Since our approach only focuses on the artefacts introduced by urban areas and considers the parallax that originates from buildings, it is not surprising that false alarms from trees are not eliminated well. However, this problem is easily overcome using the GM-PHD filter. The precision increases obviously for both detectors.

Both the proposed and 1-plane approach have high recall before or after a GM-PHD filter is performed. This is due to the regular movement happening in the absence of obscuration and because the camera rotations are less pronounced. As the vehicles always move upwards, and the background moves downwards, the problem with the proposed approach (mentioned in Section 5.4.1) that causes many mis-detections does not occur.

Several tracking results are shown in Figure 5.9. The tracking process considers using only

Number of Detections

Method	Precision	Recall
Multi-plane Background	0.602	0.884
1-plane Background	0.464	0.899

GM-PHD Refined Detections

Method	Precision	Recall
Multi-plane Background	0.776	0.850
1-plane Background	0.638	0.875

The multi-plane background is the proposed approach. The 1-plane Background is similar to what was proposed in Chapter 2.

TABLE 5.3: The experiment result on the real outdoor video using our approach.

the detections (with no appearance information[8]). Missing detections and multiple detections per object appear to be the primary reasons for the observed failures. Note that Figure 5.9 (e) shows a case in which the tracks become confused as a result of two nearby objects generating merged detections. However, Figure 5.9 (d) shows a case in which the tracks do not become confused. This problem is totally dependent on the appearance and distance between two objects.

[8]Such appearance information could be readily added.

(a) Tracking results from Frame #10 to #160.

(b) Tracking results from Frame #325 to #475.

(c) Tracking results from Frame #425 to #550.

(d) Tracking results from Frame #775 to #955.

(e) Tracking results from Frame #1075 to #1165.

(f) Tracking results from Frame #1200 to #1310.

FIGURE 5.9: Selected tracking results using proposed detector and Gaussian mixture probabilistic hypothesis density (GM-PHD) filter on the outdoor dataset.

5.5 Conclusions

Moving object detection from a moving camera by mosaic background subtraction can be challenging in the urban areas when pronounced parallax occurs. In this chapter, we proposed an algorithm for such case. The approach is still based on a background subtraction algorithm. Moreover we consider that the scene comprises multi-planes. VO and GME are used to identify and compensating the motion for each plane. The background modelling is therefore more robust to the parallax. It has been shown that this approach can outperform the 1-plane assumption for building the background model. The detections are obtained straightforwardly and they are projected to a unified space with considering the VO information, such that a scaled constant velocity model can be used for tracking. Experiments proved that this approach can deal with pronounced parallax in similar urban areas. Finally, the detections can be processed

by a GM-PHD filter. Some false alarms can be further eliminated and reasonable tacks can be produced.

Chapter 6

Conclusions

6.1 Summary

Detecting and tracking moving objects are fundamental tasks in the analysis of surveillance video. Many methods are potentially applicable. Some are well suited to static cameras. Some work well with moving cameras. As the motivation of this thesis is to process data from a large area, the focus here is on an airborne camera. If the camera is high relative to the depth of the scene, moving object detection can be solved by first estimating the camera motion using the affine/homography transformation and then estimating a background mosaic. Background subtraction can then be used to detect objects.

Chapter 2 firstly described and compared several existing feature-based and pixel-based global motion estimation (GME) methods. In order to achieve higher accuracy, we focused on the pixel-based algorithms where camera motion estimation can be posed as an optimisation problem. We proposed a robust and adjustable cost function, the student-t cost function. A parameter auto-tuning method has been introduced for trading off efficiency and robustness as a video sequence is processed.

The particle filter is a powerful tool for addressing non-linear problems in the field of state estimation and object tracking. Chapter 3 had an intensive discussion about particle filtering which led to estimating the state of camera in Chapter 4. There are several techniques that improve sampling efficiency and can thereby reduce computational cost. Two such techniques are: the use of a near-optimal proposal; and Rao-Blackwellisation. While many papers have discussed these approaches in isolation, there are only a few previous examples of research into combining the two methods. More specifically, a general combined implementation strategy that can cater for the correlated process noise that arises has, previously, been mentioned, but without comparisons and discussions in particular scenarios. This chapter presented such an approach and also considered several scenarios to show when, in terms of estimation accuracy and computational cost, it was appropriate to use each technique in isolation or both together.

In Chapter 4, we came back to the topic of analysing the surveillance videos. When an airborne camera is at low altitude, there will often be pronounced parallax in the video. We

proposed a solution to the visual odometry (VO) problem, i.e., to the problem of extracting the position orientation of the camera and the position of the landmarks in such scenarios. Using the aforementioned particle filter in Chapter 3 (which uses the optimal proposal and Rao-Blackwellisation twice, that we call RB^2-PF), the camera states can be estimated with fewer particles then the famous FastSLAM 2.0. The thesis shows that it is possible to reduce the runtime in such problems: doing so is significant for real-time problems.

Using RB^2-PF based VO as a key component, a moving object detection algorithm that can cater for pronounced parallax was proposed in Chapter 5. Instead of considering all the objects in the scene on one plane (as what we have done in Chapter 2), we assumes the landmarks from VO are on multiple planes. We can therefore calculate the GME for each plane and better maintains the background model. Experiments are conducted using a simulation of the city and real data from an outdoor benchmark. The proposed approach considerably reduces the false alarm rate, and provides useful sensitivity, relative to existing approaches.

6.2 Future Work

We suggest the future research avenues in relation to this thesis and in the field of moving object detection from a moving camera and particle filtering as follows.

- Proposed student-t cost function has been proved to be suitable for Global Motion Estimation problem which was a necessary pre-processing step for moving object detection. However, the proposed parameter estimation method was exhaustive. The method was not efficient when the content of the image changed frequently. As the quantity of the outliers can be directly estimated from a segment of the video. It is possible to estimate the suitable parameters by machine learning, such as neural network.
- A well-known issue of monocular visual odometry is that the scale of the map cannot be estimated due to the lack of depth information from single camera. Using stereo camera or auxiliary inertial measurement unit can fix the problem. While considering the cases that the control parameters are unknown, a near-constant velocity model can be used to model the camera dynamics and the proposed model can be easily applied. But the measurement model should be adjusted accordingly. Therefore, the future work can be: extending the proposed approach that can fuse different sensors.
- A GM-PHD filter was used to track multiple targets in this thesis which only considered the positions. When there are multiple targets that are close to each other, data association becomes challenging. However, for videos, the appearances (e.g., size or colour) of the targets are also obtained without extra cost. The appearances can be fused into state-space and improve the accuracy of data association. We suggest this as a future work.
- Chapter 2 and Chapter 5 proposed ways to detect moving objects when the scene was flat or pronounced parallax occurred. The background subtraction based algorithm exploits temporal information. Thus it detects the moving objects which may not be our interest

and it cannot detect the objects that are stationary. As mentioned in the summary of previous work on deep learning, deep learning can exploit the appearance of the objects and has the potential to give rise to huge improvements in accuracy. Future work will therefore combine the unsupervised approaches that are proposed in this thesis with such deep learning methods. The intent is that the approaches developed in this thesis will be used to identify the temporal information associated with each of many regions. The deep learning will then process the resulting sequences to yield accurate detections.

Bibliography

[1] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for uav tracking," in *European Conference on Computer Vision*, pp. 445–461, Springer, 2016.

[2] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[3] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, (Hong Kong, China), May 2014.

[4] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–I, IEEE, 2001.

[5] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[10] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[12] R. Girshick, "Fast R-CNN," *arXiv preprint arXiv:1504.08083*, 2015.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[16] Y. Zhou and S. Maskell, "RB^2-PF : A novel filter-based monocular visual odometry algorithm," in *20th International Conference on Information Fusion, FUSION*, 2017.

[17] Y. Zhou and S. Maskell, "Moving object detection using background subtraction for a moving camera with pronounced parallax," in *11th Symposium Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, 2017.

[18] C. Y. Liu, Y. Zhou, F. de Melo, and S. Maskell, "Probabilistic graphical detector fusion for localization of faces and facial parts," in *2014 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pp. 1–6, Oct 2014.

[19] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision.," in *IJCAI*, vol. 81, pp. 674–679, 1981.

[20] O. Barnich and M. Van Droogenbroeck, "ViBe: A universal background subtraction algorithm for video sequences," *IEEE Transactions on Image processing*, vol. 20, no. 6, pp. 1709–1724, 2011.

[21] S. Saha and F. Gustafsson, "Particle filtering with dependent noise processes," *IEEE Transactions on Signal Processing*, vol. 60, no. 9, pp. 4497–4508, 2012.

[22] M. Montemerlo and S. Thrun, *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*, vol. 27. Springer, 2007.

[23] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera slam," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[24] J. Civera, A. J. Davison, and J. M. Montiel, "Inverse depth parametrization for monocular SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[25] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2548–2555, IEEE, 2011.

[26] B. Wang and P. Dudek, "A fast self-tuning background subtraction algorithm," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 395–398, 2014.

[27] P. Azzari, L. Di Stefano, and A. Bevilacqua, "An effective real-time mosaicing algorithm apt to detect motion through background subtraction using a PTZ camera," in *IEEE Conference on Advanced Video and Signal Based Surveillance*, pp. 511–516, IEEE, 2005.

[28] D. D. Doyle, A. L. Jennings, and J. T. Black, "Optical flow background subtraction for real-time PTZ camera object tracking," in *IEEE International Instrumentation and Measurement Technology Conference, I2MTC*, pp. 866–871, IEEE, 2013.

[29] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[30] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," *International journal of computer vision*, vol. 56, no. 3, pp. 221–255, 2004.

[31] C. Liu, *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, Citeseer, 2009.

[32] Y. Sheikh, O. Javed, and T. Kanade, "Background subtraction for freely moving cameras," in *IEEE 12th International Conference on Computer Vision*, pp. 1219–1225, IEEE, 2009.

[33] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," *European conference on computer vision*, pp. 282–295, 2010.

[34] T. Lim, B. Han, and J. H. Han, "Modeling and segmentation of floating foreground and background in videos," *Pattern Recognition*, vol. 45, no. 4, pp. 1696–1706, 2012.

[35] S. Araki, T. Matsuoka, N. Yokoya, and H. Takemura, "Real-time tracking of multiple moving object contours in a moving camera image sequence," *IEICE Transactions on Information and Systems*, vol. 83, no. 7, pp. 1583–1591, 2000.

[36] P. Kent, S. Maskell, O. Payne, S. Richardson, and L. Scarff, "Robust background subtraction for automated detection and tracking of targets in wide area motion imagery," in *SPIE Security+ Defence*, pp. 85460Q–85460Q, International Society for Optics and Photonics, 2012.

[37] S. W. Kim, K. Yun, K. M. Yi, S. J. Kim, and J. Y. Choi, "Detection of moving objects with a moving camera using non-panoramic background model," *Machine vision and applications*, vol. 24, no. 5, pp. 1015–1028, 2013.

[38] C.-T. Hsu and Y.-C. Tsan, "Mosaics of video sequences with moving objects," *Signal Processing: Image Communication*, vol. 19, no. 1, pp. 81–98, 2004.

[39] F. Vella, A. Castorina, M. Mancuso, and G. Messina, "Digital image stabilization by adaptive block motion vectors filtering," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 3, pp. 796–801, 2002.

[40] Q. Wei, H. J. Zhang, and Y. Z. Zhong, "A pre-analysis method for robust global motion estimation," in *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, vol. 2, pp. 625–628, IEEE, 1999.

[41] T. Botterill, S. Mills, and R. Green, "Real-time aerial image mosaicing," in *Image and Vision Computing New Zealand (IVCNZ), 2010 25th International Conference of*, pp. 1–8, IEEE, 2010.

[42] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, pp. I–652, IEEE, 2004.

[43] Y.-M. Chen and I. V. Bajic, "A joint approach to global motion estimation and motion segmentation from a coarsely sampled motion vector field," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 9, pp. 1316–1328, 2011.

[44] M. Haller, A. Krutz, and T. Sikora, "Robust global motion estimation using motion vectors of variable size blocks and automatic motion model selection," in *2010 IEEE International Conference on Image Processing*, pp. 737–740, IEEE, 2010.

[45] B. Zitova and J. Flusser, "Image registration methods: a survey," *Image and vision computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[46] R. Szeliski, "Image alignment and stitching: A tutorial," *Foundations and Trends® in Computer Graphics and Vision*, vol. 2, no. 1, pp. 1–104, 2006.

[47] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2, pp. 1150–1157, Ieee, 1999.

[48] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer vision–ECCV 2006*, pp. 404–417, 2006.

[49] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE international conference on*, pp. 2564–2571, IEEE, 2011.

[50] M. Haller, A. Krutz, and T. Sikora, "Evaluation of pixel-and motion vector-based global motion estimation for camera motion characterization," in *2009 10th Workshop on Image Analysis for Multimedia Interactive Services*, pp. 49–52, IEEE, 2009.

[51] C. V. Stewart, "Robust parameter estimation in computer vision," *SIAM review*, vol. 41, no. 3, pp. 513–537, 1999.

[52] M. J. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields," *Computer vision and image understanding*, vol. 63, no. 1, pp. 75–104, 1996.

[53] K.-H. Yap, Y. He, Y. Tian, and L.-P. Chau, "A nonlinear-norm approach for joint image registration and super-resolution," *IEEE Signal Processing Letters*, vol. 16, no. 11, pp. 981–984, 2009.

[54] K. Arya, P. Gupta, P. K. Kalra, and P. Mitra, "Image registration using robust m-estimators," *Pattern Recognition Letters*, vol. 28, no. 15, pp. 1957–1968, 2007.

[55] M. J. Black and P. Anandan, "A framework for the robust estimation of optical flow," in *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pp. 231–236, IEEE, 1993.

[56] A. Marrs, S. Maskell, and Y. Bar-Shalom, "Expected likelihood for tracking in clutter with particle filters," in *AeroSense 2002*, pp. 230–239, International Society for Optics and Photonics, 2002.

[57] D. Gerogiannis, C. Nikou, and A. Likas, "The mixtures of student's t-distributions as a robust framework for rigid registration," *Image and Vision Computing*, vol. 27, no. 9, pp. 1285–1294, 2009.

[58] Z. Zhou, J. Zheng, Y. Dai, Z. Zhou, and S. Chen, "Robust non-rigid point set registration using student's-t mixture model," *PloS one*, vol. 9, no. 3, p. e91381, 2014.

[59] O. Sorkine-Hornung and M. Rabinovich, "Least-squares rigid motion using svd," *Computing*, vol. 1, p. 1, 2017.

[60] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992.

[61] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[62] J. Shi *et al.*, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pp. 593–600, IEEE, 1994.

[63] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2, pp. 1508–1515, IEEE, 2005.

[64] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*, pp. 430–443, Springer, 2006.

[65] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, "Adaptive and generic corner detection based on the accelerated segment test," in *European conference on Computer vision*, pp. 183–196, Springer, 2010.

[66] J. P. Lewis, "Fast template matching," in *Vision interface*, vol. 95, pp. 15–19, 1995.

[67] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," in *Readings in computer vision*, pp. 726–740, Elsevier, 1987.

[68] P. H. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Computer vision and image understanding*, vol. 78, no. 1, pp. 138–156, 2000.

[69] R. Collins, X. Zhou, and S. K. Teh, "An open source tracking testbed and evaluation web site," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, vol. 35, 2005.

[70] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid methods in image processing," *RCA engineer*, vol. 29, no. 6, pp. 33–41, 1984.

[71] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, no. 1-10, p. 4, 2001.

[72] J. Fox *et al.*, *Robust regression, Appendix to An R and S-PLUS Companion to Applied Regression*. 2002.

[73] T. Pock, M. Urschler, C. Zach, R. Beichel, and H. Bischof, "A duality based algorithm for tv-l1-optical-flow image registration," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 511–518, Springer, 2007.

[74] P. J. Huber, "Robust statistics," in *International Encyclopedia of Statistical Science*, pp. 1248–1251, Springer, 2011.

[75] J. W. Tukey, *Exploratory data analysis*. Reading, Mass., 1977.

[76] S. Zhang, C. Tepedelenlioğlu, M. K. Banavar, and A. Spanias, "Max consensus in sensor networks: Non-linear bounded transmission and additive noise," *IEEE Sensors Journal*, vol. 16, no. 24, pp. 9089–9098, 2016.

[77] B.-N. Vo and W.-K. Ma, "The gaussian mixture probability hypothesis density filter," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4091–4104, 2006.

[78] R. G. Brown and P. Y. Hwang, "Introduction to random signals and applied kalman filtering: with MATLAB exercises and solutions," *Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions, by Brown, Robert Grover.; Hwang, Patrick YC New York: Wiley, c1997.*, vol. 1, 1997.

[79] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *AeroSense'97*, pp. 182–193, International Society for Optics and Photonics, 1997.

[80] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on signal processing*, vol. 50, no. 2, pp. 174–188, 2002.

[81] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," *Handbook of Nonlinear Filtering*, vol. 12, no. 656-704, p. 3, 2009.

[82] R. Van Der Merwe, A. Doucet, N. De Freitas, and E. Wan, "The unscented particle filter," in *NIPS*, vol. 2000, pp. 584–590, 2000.

[83] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pp. 153–158, Ieee, 2000.

[84] D. Guo, X. Wang, and R. Chen, "New sequential monte carlo methods for nonlinear dynamic systems," *Statistics and Computing*, vol. 15, no. 2, pp. 135–147, 2005.

[85] S. Maskell, M. Briers, R. Wright, and P. Horridge, "Tracking using a radar and a problem specific proposal distribution in a particle filter," *IEE Proceedings-Radar, Sonar and Navigation*, vol. 152, no. 5, pp. 315–322, 2005.

[86] C. R. Rao, "Information and accuracy attainable in the estimation of statistical parameters," *Bull Calcutta. Math. Soc.*, vol. 37, pp. 81–91, 1945.

[87] D. Blackwell, "Conditional expectation and unbiased sequential estimation," *The Annals of Mathematical Statistics*, pp. 105–110, 1947.

[88] P.-J. Nordlund and F. Gustafsson, "Sequential monte carlo filtering techniques applied to integrated navigation systems," in *American Control Conference, 2001. Proceedings of the 2001*, vol. 6, pp. 4375–4380, IEEE, 2001.

[89] G. Casella and C. P. Robert, "Rao-blackwellisation of sampling schemes," *Biometrika*, vol. 83, no. 1, pp. 81–94, 1996.

[90] T. Schön, R. Karlsson, and F. Gustafsson, "The marginalized particle filter in practice," in *2006 IEEE Aerospace Conference, Big Sky, MT, USA, March, 2006*, 2006.

[91] G. Hendeby, R. Karlsson, and F. Gustafsson, "The rao-blackwellized particle filter: a filter bank implementation," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, no. 1, p. 1, 2010.

[92] A. Doucet, N. De Freitas, K. Murphy, and S. Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pp. 176–183, Morgan Kaufmann Publishers Inc., 2000.

[93] T. Schon, F. Gustafsson, and P.-J. Nordlund, "Marginalized particle filters for mixed linear/nonlinear state-space models," *IEEE Transactions on Signal Processing*, vol. 53, no. 7, pp. 2279–2289, 2005.

[94] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 53–82, 2010.

[95] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Aaai/iaai*, pp. 593–598, 2002.

[96] M. R. Morelande, C. M. Kreucher, and K. Kastella, "A bayesian approach to multiple target detection and tracking," *IEEE Transactions on Signal Processing*, vol. 55, no. 5, pp. 1589–1604, 2007.

[97] P. Closas, C. Fernandez-Prades, and J. A. Fernandez-Rubio, "A bayesian approach to multipath mitigation in gnss receivers," *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, no. 4, pp. 695–706, 2009.

[98] A. Giremus, E. Grivel, J. Grolleau, and M. Najim, "A rao-blackwellized particle filter for joint channel/symbol estimation in mc-ds-cdma systems," *IEEE Transactions on Communications*, vol. 58, no. 8, pp. 2292–2304, 2010.

[99] E. E. Tsakonas, N. D. Sidiropoulos, and A. Swami, "Optimal particle filters for tracking a time-varying harmonic or chirp signal," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4598–4610, 2008.

[100] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking: dynamic models," in *AeroSense 2000*, pp. 212–235, International Society for Optics and Photonics, 2000.

[101] J. S. Liu, *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

[102] J. Carpenter, P. Clifford, and P. Fearnhead, "Improved particle filter for nonlinear problems," *IEE Proceedings-Radar, Sonar and Navigation*, vol. 146, no. 1, pp. 2–7, 1999.

[103] A. Doucet, N. De Freitas, and N. Gordon, "An introduction to sequential monte carlo methods," in *Sequential Monte Carlo Methods in Practice*, pp. 3–14, Springer, 2001.

[104] J. V. Candy, "Bootstrap particle filtering," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 73–85, 2007.

[105] M. Briers, S. R. Maskell, and R. Wright, "A rao-blackwellised unscented kalman filter," *Info. Fusion*, pp. 8–11, 2003.

[106] S. Maskell, M. Rollason, N. Gordon, and D. Salmond, "Efficient particle filtering for multiple target tracking with application to tracking in structured images," *Image and Vision Computing*, vol. 21, no. 10, pp. 931–939, 2003.

[107] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.

[108] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on aerospace and electronic systems*, vol. 39, no. 4, pp. 1333–1364, 2003.

[109] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on robotics and automation*, vol. 17, no. 3, pp. 229–241, 2001.

[110] C. Kim, R. Sakthivel, and W. K. Chung, "Unscented FastSLAM: a robust and efficient solution to the slam problem," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 808–820, 2008.

[111] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pp. 225–234, IEEE, 2007.

[112] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[113] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2320–2327, IEEE, 2011.

[114] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision*, pp. 834–849, Springer, 2014.

[115] H. Strasdat, J. Montiel, and A. J. Davison, "Real-time monocular SLAM: Why filter?," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2657–2664, IEEE, 2010.

[116] L. Kneip, M. Chli, R. Siegwart, *et al.*, "Robust real-time visual odometry with a single camera and an imu," in *BMVC*, pp. 1–11, 2011.

[117] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular slam with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, pp. 796–803, April 2017.

[118] E. Eade and T. Drummond, "Scalable monocular SLAM," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, pp. 469–476, IEEE, 2006.

[119] T. Schön, R. Karlsson, D. Törnqvist, and F. Gustafsson, "A framework for simultaneous localization and mapping utilizing model structure," in *Proceedings of the 10th International Conference on Information Fusion*, pp. 1–8, IEEE, 2007.

[120] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[121] C. Kim, H. Kim, and W. K. Chung, "Exactly rao-blackwellized unscented particle filters for slam," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3589–3594, IEEE, 2011.

[122] S. J. Julier, "The scaled unscented transformation," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, vol. 6, pp. 4555–4559, IEEE, 2002.

[123] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using fastslam," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2, pp. 1985–1991, IEEE, 2003.

[124] D. Hähnel, S. Thrun, B. Wegbreit, and W. Burgard, "Towards lazy data association in SLAM," in *Robotics Research. The Eleventh International Symposium*, pp. 421–431, Springer, 2005.

[125] J. Neira and J. D. Tardós, "Data association in stochastic mapping using the joint compatibility test," *IEEE Transactions on robotics and automation*, vol. 17, no. 6, pp. 890–897, 2001.

[126] S. W. Shepperd, "Quaternion from rotation matrix.[four-parameter representation of coordinate transformation matrix," *Journal of guidance and control*, 1978.

[127] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[128] V. Aidala and S. Hammel, "Utilization of modified polar coordinates for bearings-only tracking," *IEEE Transactions on Automatic Control*, vol. 28, no. 3, pp. 283–294, 1983.

[129] S. Arulampalam and B. Ristic, "Comparison of the particle filter with range parameterized and modified polar ekf's for angle-only tracking," in *Proc. Spie*, vol. 4048, pp. 288–299, 2000.

[130] O. Miksik and K. Mikolajczyk, "Evaluation of local detectors and descriptors for fast feature matching," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, pp. 2681–2684, IEEE, 2012.

[131] S. S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Artech House Publishers, 1999.

[132] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "Svo: Semi-direct visual odometry for monocular and multi-camera systems," *IEEE Transactions on Robotics and Automation, to appear.*, 2017.

[133] Y. Jin, L. Tao, H. Di, N. I. Rao, and G. Xu, "Background modeling from a free-moving camera by multi-layer homography algorithm," in *15th IEEE International Conference on Image Processing, ICIP*, pp. 1572–1575, IEEE, 2008.

[134] D. Zamalieva and A. Yilmaz, "Background subtraction for the moving camera: A geometric approach," *Computer Vision and Image Understanding*, vol. 127, pp. 73–85, 2014.

[135] S. Kim, D. W. Yang, and H. W. Park, "A disparity-based adaptive multihomography method for moving target detection based on global motion compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 8, pp. 1407–1420, 2016.

[136] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR*, vol. 2, pp. 246–252, IEEE, 1999.

[137] E. J. Griffith, C. Mishra, J. F. Ralph, and S. Maskell, "A system for the generation of synthetic wide area aerial surveillance imagery," *Simulation Modelling Practice and Theory*, vol. 84, pp. 286 – 308, 2018.

[138] R. E. Kalman *et al.*, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[139] M. S. Grewal, "Kalman filtering," in *International Encyclopedia of Statistical Science*, pp. 705–708, Springer, 2011.

[140] J. J. LaViola, "A comparison of unscented and extended kalman filtering for estimating quaternion motion," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 3, pp. 2435–2440, IEEE, 2003.

[141] S. Blackman and R. Popoli, "Design and analysis of modern tracking systems(book)," *Norwood, MA: Artech House, 1999.*, 1999.

[142] Y. Bar-Shalom and E. Tse, "Tracking in a cluttered environment with probabilistic data association," *Automatica*, vol. 11, no. 5, pp. 451–460, 1975.

Appendix A

Kalman Filter and Extensions

A.1 Kalman Filter

A.1.1 Kalman Filter Definition

Since the Kalman filter is a well-known solution to linear Gaussian problems, only the core implementations (as follows) and the derivation of the Kalman gain (in the next subsection) will be presented. The detailed descriptions and derivations can be found in [138] or [139].

The prediction phase is as follows:

$$\hat{x}_{t|t-1} = F\hat{x}_{t-1|t-1} \tag{A.1}$$

$$P_{t|t-1} = F \cdot P_{t-1|t-1} \cdot F^T + Q. \tag{A.2}$$

The updating phase is as follows:

$$S_t = H \cdot P_{t|t-1} \cdot H^T + R \tag{A.3}$$

$$K_t = P_{t|t-1}H^T S_t^{-1} \tag{A.4}$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - H_t\hat{x}_{t|t-1}) \tag{A.5}$$

$$P_{t|t} = (I - K_t H)P_{t|t-1}. \tag{A.6}$$

A.1.2 Derivation of the Kalman Gain

The main idea of the Kalman filter is to find the Kalman gain, K_t, to minimise the posterior uncertainty covariance $P_{t|t}$, given the uncertainties of states and observations. The minimisation is equivalent to minimising the variance of each element on the main diagonal (the trace) of $P_{t|t}$, which is the task of the following derivation.

For preparation, the posterior uncertainty covariance is presented as follows:

$$P_{t|t} = \left[||x_t - \hat{x}_{t|t}||^2\right], \tag{A.7}$$

where x_t is the true state, and $\hat{x}_{t|t}$ is the posterior state estimation of time t.

By substituting the state prediction $\hat{x}_{t|t-1}$, true measurement $H_t x_t$, measurement noise ϵ_t, and Kalman gain K_t into A.7, we obtain the following:

$$\begin{aligned}
P_{t|t} =& cov\left(x_t - \left[\hat{x}_{t|t-1} + K_t(H_t x_t + \epsilon_t - H_t \hat{x}_{t|t-1})\right]\right) && \text{(A.8)}\\
=& cov\left[(I - K_t H_t)(x_t - \hat{x}_{t|t-1}) - K_t \epsilon_t\right] && \text{(A.9)}\\
=& cov\left[(I - K_t H_t)(x_t - \hat{x}_{t|t-1})\right] + cov\left[K_t \epsilon_t\right] && \text{(A.10)}\\
=& cov(I - K_t H_t) cov(x_t - \hat{x}_{t|t-1})(I - K_t H_t)^T + K_t cov(\epsilon_t) K_t^T. && \text{(A.11)}
\end{aligned}$$

If the problem is modelled precisely (where the initial state and covariance are accurate, the process noises and measurement noises follow the true distribution), we should obtain the following:

$$\begin{aligned}
P_{t|t} =& (I - K_t H_t) P_{t|t-1} (I - K_t H_t)^T + K_t R_t K_t^T && \text{(A.12)}\\
=& P_{t|t-1} - K_t H_t P_{t|t-1} - P_{t|t-1} H_t^T K_t^T + K_t (H_t P_{t|t-1} H_t^T) K_t^T && \text{(A.13)}\\
=& P_{t|t-1} - K_t H_t P_{t|t-1} - P_{t|t-1} H_t^T K_t^T + K_t S_t K_t^T. && \text{(A.14)}
\end{aligned}$$

To minimise the trace of $P_{t|t}$, we make the derivative of $P_{t|t}$ with respect to K_t zero, that is:

$$\frac{\partial tr(P_{t|t})}{\partial K_t} = -2(H_t P_{\|t-1})^T + 2K_t S_t = 0. \tag{A.15}$$

By re-arranging (A.15), the Kalman gain can be calculated as follows:

$$K_t = P_{t|t-1} H_t^T S_t^{-1}. \tag{A.16}$$

A.2 Extended Kalman Filter (EKF)

The EKF uses the first derivative of the non-linear functions to transform the non-linear problem into an approximated linear problem that can be processed by the Kalman filter. The procedures are as follows:

The prediction phase is:

$$\hat{x}_{t|t-1} = f(\hat{x}_{t-1|t-1}) \tag{A.17}$$

$$P_{t|t-1} = \dot{F}_{k-1} \cdot P_{t-1|t-1} \cdot \dot{F}_{k-1}^T + Q. \tag{A.18}$$

The first derivative of the dynamic is:

$$\dot{F}_{t-1} = \left.\frac{\partial f(.)}{\partial x}\right|_{\hat{x}_{t-1|t-1}}. \tag{A.19}$$

The updating phase is:

$$S_t = \dot{H}_t \cdot P_{t|t-1} \cdot \dot{H}_t^T + R \tag{A.20}$$

$$K_t = P_{t|t-1}\dot{H}_t^T S_t^{-1} \tag{A.21}$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - h(\hat{x}_{t|t-1})) \tag{A.22}$$

$$P_{t|t} = (I - K_t\dot{H}_t)P_{t|t-1}. \tag{A.23}$$

The first derivative of the measurement model is:

$$\dot{H}_t = \left.\frac{\partial f(.)}{\partial x}\right|_{\hat{x}_{t|t-1}}. \tag{A.24}$$

A.3 Unscented Kalman Filter

The UKF uses the unscented transformation to calculate the linearised mean and uncertainties. The main idea of an unscented transformation is to sample the uncertainty in previous time with a number (usually $2D+1$ where D is the dimension of the state) of sigma points and then propagate the sigma points through the non-linear transitions. Finally, the Gaussian distribution that is encoded by the sigma points can form the mean and covariance for updating a Kalman filter. The detailed explanation and application can be found in [83]. The implementation of the algorithm is presented as follows.

After obtaining the state and uncertainty from the previous time step, we start by generating the sigma points, $\chi_{t-1|t-1}^{(i)}$ (where i is the index of the sigma point), and their weights, $W_{\mu,t-1|t-1}^{(i)}$ and $W_{\sigma,t-1|t-1}^{(i)}$. An augmented state and covariance can be constructed for simpler implementation:

$$x_{t-1|t-1}^a = \begin{bmatrix} x_{t-1|t-1} \\ 0, \end{bmatrix} \tag{A.25}$$

$$P^a_{t-1|t-1} = \begin{bmatrix} P_{t-1|t-1} & 0 \\ 0 & \mathrm{Q}, \end{bmatrix} \tag{A.26}$$

$$\begin{aligned}
\chi^{(i)}_{t-1|t-1} &= x^a_{t-1|t-1}, && i = 1 \\
\chi^{(i)}_{t-1|t-1} &= x^a_{t-1|t-1} + \left(\sqrt{(L+\lambda)P^a_{t-1|t-1}}\right), && i = 2, ..., D+1 \\
\chi^{(i)}_{t-1|t-1} &= x^a_{t-1|t-1} - \left(\sqrt{(L+\lambda)P^a_{t-1|t-1}}\right), && i = D+2, ..., 2D+1,
\end{aligned} \tag{A.27}$$

where the root square $\sqrt{(L+\lambda)P^a_{t-1|t-1}}$ is usually computed via Cholesky decomposition, assuming D is the dimension number of the augmented state, $x^a_{t-1|t-1}$:

$$\begin{aligned}
W^{(i)}_{\mu,t-1|t-1} &= \frac{\lambda}{L+\lambda} && i = 1 \\
W^{(i)}_{\mu,t-1|t-1} &= \frac{1}{2(L+\lambda)} && i = 2, ..., 2D+1 \\
W^{(i)}_{\sigma,t-1|t-1} &= \frac{\lambda}{L+\lambda} + 1 - \alpha^2 + \beta && i = 1 \\
W^{(i)}_{\sigma,t-1|t-1} &= \frac{1}{2(L+\lambda)} && i = 2, ..., 2D+1.
\end{aligned} \tag{A.28}$$

The prediction phase is described as follows.

Propagating the sigma points using the transmission function, we obtain:

$$\dot{\chi}^{(i)}_{t|t-1} = f(\chi^{(i)}_{t-1|t-1}). \tag{A.29}$$

Calculating the prediction with the weighted sigma points, we obtain:

$$\hat{x}_{t|t-1} = \sum_{i=1}^{i=2D+1} W^{(i)}_{\mu,t-1|t-1}\dot{\chi}^{(i)}_{t|t-1} \tag{A.30}$$

$$P_{t|t-1} = \sum_{i=1}^{i=2D+1} W^{(i)}_{\sigma,t-1|t-1}\left[\dot{\chi}^{(i)}_{t|t-1} - \hat{x}_{t|t-1}\right]\left[\dot{\chi}^{(i)}_{t|t-1} - \hat{x}_{t|t-1}\right]^T. \tag{A.31}$$

Before the measurement update, we need to augment the measurement uncertainty. Again, sigma points, $\chi^{(i)}_{t|t-1}$, should be generated using the equations in (A.27) with which the augmented matrices $x^a_{t-1|t-1}$ and $P^a_{t-1|t-1}$ are substituted by (A.32) and (A.33).[1] Their weights ($W^{(i)}_{\mu,t|t-1}$

[1]It is possible to use the sigma points $\dot{\chi}^{(i)}_{t|t-1}$ instead of generating new $\chi^{(i)}_{t|t-1}$. If so, the weights should be recalculated that considers the measurement uncertainty joins the augmented matrix. We consider them separately here, since it is easier to be compatible to the case in which either the dynamic model or measurement model is not non-linear, which can be handled by a linear filter.

and $W^{(i)}_{\sigma,t|t-1}$) are calculated using the equations in (A.28). Note that D could change, as it is the dimension number of the new augmented state, $x^a_{t|t-1}$.

$$x^a_{t|t-1} = \begin{bmatrix} \hat{x}_{t|t-1} \\ 0, \end{bmatrix} \tag{A.32}$$

$$P^a_{t|t-1} = \begin{bmatrix} P_{t|t-1} & 0 \\ 0 & \mathrm{R} \end{bmatrix}. \tag{A.33}$$

The updating phase is described as follows.

Projecting the sigma points to the observation space, we have:

$$\dot{z}^{(i)}_{t|t-1} = h(\chi^{(i)}_{t|t-1}). \tag{A.34}$$

Calculating the predicted measurement states, covariance, and cross-covariance matrices with the weighted sigma points, we obtain:

$$\hat{z}_{t|t-1} = \sum_{i=1}^{i=2D+1} W^{(i)}_{\mu,t-1} \dot{z}^{(i)}_{t|t-1} \tag{A.35}$$

$$P_{zz} = \sum_{i=1}^{i=2D+1} W^{(i)}_{\sigma,t-1} \left[\dot{z}^{(i)}_{t|t-1} - \hat{z}_{t|t-1}\right] \left[\dot{z}^{(i)}_{t|t-1} - \hat{z}_{t|t-1}\right]^T \tag{A.36}$$

$$P_{xz} = \sum_{i=1}^{i=2D+1} W^{(i)}_{\sigma,t-1} \left[\dot{\chi}^{(i)}_{t|t-1} - \hat{x}_{t|t-1}\right] \left[\dot{z}^{(i)}_{t|t-1} - \hat{z}_{t|t-1}\right]^T. \tag{A.37}$$

Finally, we maintain the ordinary Kalman filter update processes, as follows:

$$K_t = P_{xz} P_{zz}^{-1} \tag{A.38}$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - \hat{z}_{t|t-1}) \tag{A.39}$$

$$P_{t|t} = P_{t|t-1} - K_t P_{zz} K_t^T. \tag{A.40}$$

It is possible to use larger augmented states (and the covariance) that include the uncertainties of the state-space, process noise, and measurement noise (i.e., merging ((A.25) and (A.32)). However, since this chapter only considers the cases in which only one of the dynamic and measurement models is linear, it will not be duplicated herein. The details can be found in [83].

Regarding time complexity, [83] has claimed that the time complexity of UKF has same order as EKF. However, it is well-known that, for a high-dimensional problem, the number of sigma points will be large. Thus, calculating the propagation for each sigma point could be costly. Furthermore, [140] compared EKF and UKF on estimating quaternion motions, and

despite the accuracy, the computational cost is vastly different. The choice is apparent when they produce similar results.

Appendix B

Data Association Methods

B.1 Nearest-Neighbour Data Association and Gating

In the real world, even for a single object tracking problem, data association is necessary when there are mis-detections or clutters. The simple and sensible method is to choose the nearest observation. However, it is possible that this observation is not relevant. One case is that the true observation is not observable, and the nearest observation is a clutter. Thus, the estimation can be wrong and could further influence the following iterations.

One way to avoid this problem is to use a gate that rules out unlike observations with prior knowledge. The process is as follows. First, the Gaussian probability for an observation, $\tilde{z}_t$, can be calculated as follows:

$$p(\tilde{z}_t) = \frac{e^{\frac{-\tilde{d}^2}{2}}}{(2\pi)^{M/2}\sqrt{|S|}}, \tag{B.1}$$

$$S = H \cdot P \cdot H^T + R, \tag{B.2}$$

$$\tilde{d} = \sqrt{(\tilde{z}_t - h(x_{t|t-1}))^T \cdot S^{-1} \cdot (\tilde{z}_t - h(x_{t|t-1}))}, \tag{B.3}$$

where S is the residual covariance matrix that appears in the Kalman filter, $|S|$ is the determinant of S, $\tilde{d}$ is a normalised distance defined by the measurement residual and the residual covariance matrix (the Mahalanobis distance), $x_{t|t-1}$ is the predicted object state, and $h(.)$ is the transformation matrix from the state-space to measurement space.

We can then use a gate, G, to check the squared normalised distance $\tilde{d}^2$. An association can be made only if:

$$\tilde{d}^2 \leq G. \tag{B.4}$$

The choice of G can be configured. It represents the number of standard deviations. The Chi-table is usually used for choosing the value given the dimension of the problem and the

anticipated reliable probability. Otherwise, [141] has described a maximum-likelihood gate, G_0 as an alternative:

$$G_0 = 2ln\left[\frac{P_D}{(1-P_D)(2\pi)^{M/2}\beta\sqrt{|S|}}\right], \tag{B.5}$$

where P_D is the probability of detection, and M is the dimension of measurement.

Furthermore , β is defined as follows.

$$\beta = \beta_{nt} + \beta_{ft}, \tag{B.6}$$

where β_{nt} is the density of new born targets, nt, and β_{ft} is the density of the false target, ft. The densities are defined by:

$$\beta_{nt} = \frac{P_{nt}}{V_C}, \tag{B.7}$$

$$\beta_{ft} = \frac{P_{ft}}{V_C}, \tag{B.8}$$

where P_{nt} and P_{ft} are the corresponding probabilities, and V_C is the measurement volume element.

B.2 Probabilistic Data Association Method

Probabilistic data association is a kind of all-neighbour data association approach that was first proposed in [142]. In this subsection, only the simplified implementation will be presented since data association is not the focus of the thesis. However, it is used in several experiments, and the idea inspired the existence score in Chapter 4.

To describe the method, we assume a linear Gaussian problem (such as in Section 3.2.1). There are N valid observations (within a gate or the maximum range of a sensor) that are detected with the additional hypothesis that all of them are clutters; therefore, there are $N+1$ hypotheses for the association. The probabilistic data association method requires the probabilities, p_{ij}, of track i can be associated with hypothesis j, which can be calculated as follows.

$$p_{ij} = \left\{ \begin{array}{ll} \frac{b}{b+\sum_{l=1}^{N}\alpha_{il}}, & j=0 \\ \frac{\alpha_{ij}}{b+\sum_{l=1}^{N}\alpha_{il}}, & 1 \leq j \leq N \end{array} \right\}, \tag{B.9}$$

where

$$b = (1 - P_D P_G)\beta(2\pi)^{M/2}\sqrt{|S_i|}, \tag{B.10}$$

$$\alpha_{ij} = P_D e^{-\frac{d_{ij}^2}{2}}, \tag{B.11}$$

and $\beta = \frac{N}{V_G}$ and V_G is the gate volume. For a two-dimensional problem (e.g., the target position is in 2D Cartesian space), V_G is computed by:

$$V_G = \pi\sqrt{|HP_{i,k|k-1}H^T + R|}G. \tag{B.12}$$

If considering this data association problem under the Kalman filter framework, the residual can be calculated:

$$\hat{z}_i = \sum_{j=1}^{N} p_{ij}\hat{z}_{ij}, \tag{B.13}$$

where

$$\hat{z}_{ij} = z_j - H \cdot x_{i,t|t-1}, \tag{B.14}$$

and y_j is the measurement of observation j.

The Kalman gain is the same as the standard gain, and the update of the mean is as follows:

$$x_{i,t|t} = x_{i,t|t-1} + K_{i,t}\hat{z}_{ij}. \tag{B.15}$$

The update of the covariance is different from the standard covariance, as the measurement considers multiple hypotheses and must be compensated for accordingly:

$$P(t|t) = p_{i0}P(t|t-1) + (1 - p_{i0}) \cdot (I - K_{i,t}H)P_{t|t-1} + K_{i,t}\left[\sum_{j=1}^{N}\left(p_{ij}\hat{z}_{ij}\hat{z}_{ij}^T\right) - \hat{z}_i\hat{z}_i^T\right]K_{i,t}^T. \tag{B.16}$$